В заключение надо отметить, что, говоря об университетской системе менеджмента качества, чаще всего указывают, что, прежде всего, это иная, новая культура. С этим стоит согласиться, только напомнить, что эта культура базируется на двух взаимодополняющих (комплиментарных) столбах: нравственном климате в коллективе и технологической платформе, базирующей современной на инновационных информационных технологиях. И в этой комплиментарности, на наш взгляд, содержится ключ к дальнейшему и устойчивому развитию университета.

Литература

1. Новая версия государственного стандарта СТБ ISO 9001-2009 / Раздаточный материал для организаций всех отраслей экономики – Минск : НП РУП БелГИСС, 2009. – 105 с.

2. О развитии в высших учебных заведениях Республики Беларусь систем управления качеством образования и приведения их в соответствие с требованиями государственных стандартов Республики Беларусь и международных стандартов. Приказ Министерства образования Республики Беларусь от 24.12.2008 № 1000 [Электронный ресурс]. - 2009. – Режим доступа: http://www.minedu.unibel.by/modules.php?op=modload&name=UpDownload&file=index&req=getit&lid=903. – Дата доступа: 01.04.2009.

3. Обеспечение качества высшего образования: европейский и белорусский опыт : сб. науч. ст.; редкол.: Е. А. Ровба [и др.]. – Гродно : ГрГУ, 2007. – С. 155–258.

4. СТБ ИСО 10015:1999 «Управление качеством. Руководящие указания по обучению».

5. *Альтшуллер, Г. С.* Поиск новых идей: от озарения к технологии: (Теория и практика решения изобретательских задач) / Г. С. Альтшуллер, Б. Л. Злотин, А. В. Зусман, В. И. Филатов. – Кишинев : Картя молдовеняскэ, 1989. – 382 с.

6. Руководство по качеству Университета Гламорган (Уэльс) // Обеспечение качества высшего образования: европейский и белорусский опыт : сб. науч. ст.; редкол.: Е. А. Ровба [и др.]. – Гродно : ГрГУ, 2007. – С. 155–258.

7. Электронный ресурс: Режим доступа http://www.wired.com/

8. *Карр, Н. Дж.* Блеск и нищета информационных технологий. Почему ИТ не являются конкурентным преимуществом / Николас Дж. Карр. – М. : Изд-во «Секрет фирмы». – 2005. - 176 с.

Лявшук Владимир Евгеньевич, старший преподаватель кафедры коммерческой деятельности и международных экономических отношений Гродненского государственного университета имени Янки Купалы, lve@tut.by

УДК 004.05

С. Н. Неборский

АДАПТАЦИЯ ГИБКИХ МЕТОДОВ РАЗРАБОТКИ ПРОГРАММНЫХ СРЕДСТВ

Рассматривается процесс разработки программных средств (ПС) на основе гибких методов. Доказывается, что применение одного конкретного гибкого метода не позволяет эффективно создать качественное ПС. Предлагается использовать совокупность гибких методов для решения данной проблемы. В частности, предлагается использовать следующие методы: гибкое моделирование, экстремальное программирование и разработку, ведомую функциями.

Введение

В последнее время значительно возрос интерес к гибким методам разработки программных средств (ПС). Все гибкие методы интуитивно близки, и выбор того метода, который оптимально решит производственную задачу, – проблема не из легких. Проанализировав существующие гибкие методы разработки, можно сделать вывод, что на сегодняшний день не существует того метода, который давал бы всегда хороший результат, независимо от условий его применения и целей проекта. В данной работе содержатся результаты исследования гибких методов. Доказано, что нельзя опираться исключительно на один метод – в большинстве случаев необходимо использовать их симбиоз.

Гибкие методы разработки ПС

Гибкая разработка ПС – это концептуальный каркас для реализации проектов программной инженерии. Гибкие методы направлены на минимизацию риска за счет разработки ПС на основе коротких повторяющихся циклов - итераций. Это попытка установить то, что первостепенная ценность – разработка непосредственно ПС, а все остальные активности (написание проектной документации, поддержание моделей архитектуры системы и т. п.) – второстепенны [1]. Следует привести причины, или условия, когда применение гибких методов является наиболее эффективным:

- отсутствуют детальные спецификации требований к системе, которую необходимо разработать;
- пользовательские требования часто изменяются;
- требования неясны и противоречивы;
- сроки реализации проекта малы,
- ограничены людские ресурсы;
- невозможно сделать анализ того, как система будет развиваться в будущем.

Отсюда вытекает и цель гибких методов разработки: ПС должно быть разработано в срок, в рамках установленного бюджета, и, самое важное, - должно быть именно таким, каким его хочет видеть заказчик [2]. Здесь уместно сослаться на манифест методологии гибкой разработки ПС, два первых постулата которого гласят [3]:

- наивысший приоритет отводится удовлетворению заказчика;
- приветствуется изменение требований.

Наиболее распространенными гибкими методами разработки являются такие, как гибкое моделирование (Agile Modeling), экстремальное программирование (XP – eXtreme Programming), динамичный метод разработки систем (DSDM - Dynamic Systems Development Method), Скрум (Scrum), разработка, ведомая функциями (FDD – Feature-Driven Development), разработка, ведомая тестами (TDD - Test-Driven Development), адаптивная разработка ПС (ASD – Adaptive Software Development). Краткая характеристика данных методов описана в таблице.

Характеристика гибких методов разработки ПС

Метод	Характеристика
Agile Modeling	Метод предназначен для создания моделей ПС в условиях изменяющихся требований
XP	Метод представляет собой большей частью набор практик, изменение требований не контролируется
Scrum	Метод сводится к управлению проектом, не предполагает комплексного подхода к созданию ПС, изменение требований рассматривается как уничтожение старой функциональности и создание новой
FDD	Опора на функции как на некоторые формализованные удобные для реализации описания требований, изменение требований допускается лишь на новом инкременте реализации ПС
DSDM	Команда наделена существенными полномочиями, реверсивность любых изменений, фиксация требований на раннем этапе разработки ПС
TDD	Преувеличение роли тестов, реализация определенного функционала лишь после того, как написан и проверен его тест
ASD	Разработка ПС - постоянный процесс адаптации к текущему состоянию проекта; предполагает, что заказчик не может точно определить своих требований, а разработчик всегда должен искать баланс между определенностью и неясностью в проекте

Адаптация гибких методов разработки ПС

Изучив каждый из приведенных гибких методов, можно сделать вывод, что не существует универсального решения проблемы эффективного создания ПС в условиях постоянно изменяющихся требований. Именно поэтому необходим некий симбиоз, некая совокупность совместно используемых методов. В качестве таких методов были выбраны гибкое моделирование (Agile Modeling), экстремальное программирование (XP) и разработка, ведомая функциями (FDD).

Когда бы ни требовалось создать архитектуру системы, имеет смысл использовать гибкое моделирование. Модели, получаемые в результате, обладают следующими характеристиками [4]:

- предоставляют позитивную ценность – т. е. кто-то должен действительно в них нуждаться;
- строго преследуют свою цель (и не более); например, если цель модели – пояснить, как связаны определенные классы в коде, то модель должна делать только это;
- понятны для тех, кому они предназначены, но не обязательно для всех;
- достаточно точны и непротиворечивы для своей целевой аудитории;
- достаточно детальны; например, для объяснения структуры группы классов отдельный набор методов и переменных может быть опущен;
- просты, насколько это возможно.

Гибкие методы делают акцент на непосредственной разработке ПС. Для эффективного ведения разработки необходим набор определенных принципов и практических приемов. Именно поэтому всегда необходимо рассматривать и считаться с идеями XP. В основе XP лежат следующие принципы [5]:

- коммуникация – всегда важно общение, в частности между разработчиком и пользователем;
- простота – разрабатываемое ПС должно быть предельно простым;
- обратная связь – необходимо предоставить возможность пользователям давать отзывы о системе как можно раньше, и насколько это возможно часто;
- требовательность к команде разработчиков, состоящей преимущественно из профессионалов.

Эти четыре принципа привели к следующим идеям, составляющим основу XP [6]:

- парное программирование;
- всегда оставаться на связи с заказчиком;
- тестирование упреждает разработку;
- короткие итерации;
- все должно быть максимально простым;
- не разрабатывать ничего лишнего, пока не возникнет непосредственное требование заказчика;
- коллективное владение проектом.

Безусловно, гибкое моделирование и XP способствуют эффективной разработке ПС. Существует даже такой метод, как XP масштаба предприятия (EXP – Enterprise XP), суть которого как раз состоит в объединении XP и гибкого моделирования [1]. Однако ни XP, ни гибкое моделирование не касаются планирования проекта. Но для заказчика ведь важно не только качество ПС, но также время его разработки и бюджет. Без хорошего планирования вероятность успеха любого проекта снижается многократно. Разработка, ведомая функциями, (FDD) добавляет необходимый элемент контроля.

Согласно FDD функция – это такое требование, реализация которого запланирована и связана с определенной активностью по его реализации. Т. е. функции связывают требования с планированием. FDD предполагает наличие жестко определенных временных рамок, которые отводятся на реализацию набора функции. Эти временные рамки, набор функций (запланированных к реализации требований), их приоритеты и определяют итерацию в FDD [7].

Таким образом, в данной работе предлагается использовать не один конкретный метод в компании для разработки ПС, а следующие методы:

- метод гибкого моделирования, что позволяет создать архитектуру ПС;
- метод XP, что позволяет эффективно разрабатывать ПС;
- метод FDD, который дает управление проектом.

Этапы разработки ПС

Применение предлагаемых для использования в данной работе методов предполагает наличие следующих этапов реализации ПС:

- инициация – создание общей картины решаемой задачи, проектирование и планирование;
- стадия итераций – процесс реализации и поставки работающего ПС, при котором на каждой итерации реализуются новые требования и возможно изменяются старые;
- сопровождение – исправление обнаруживаемых ошибок и реализация новых требований, возникших после поставки ПС.

На рисунке 1 приведена взаимосвязь предлагаемых этапов разработки.

Итеративной разработке ПС предшествует этап инициации процесса. На этом этапе создается общая картина решаемой задачи. Выполняется анализ бизнес-требований заказчика. Разработчики уясняют проблему, которая стоит перед заказчиком. Определяется, как разрабатываемое ПС позволит решить эту проблему. При этом учитывается опыт реализации предыдущих проектов.

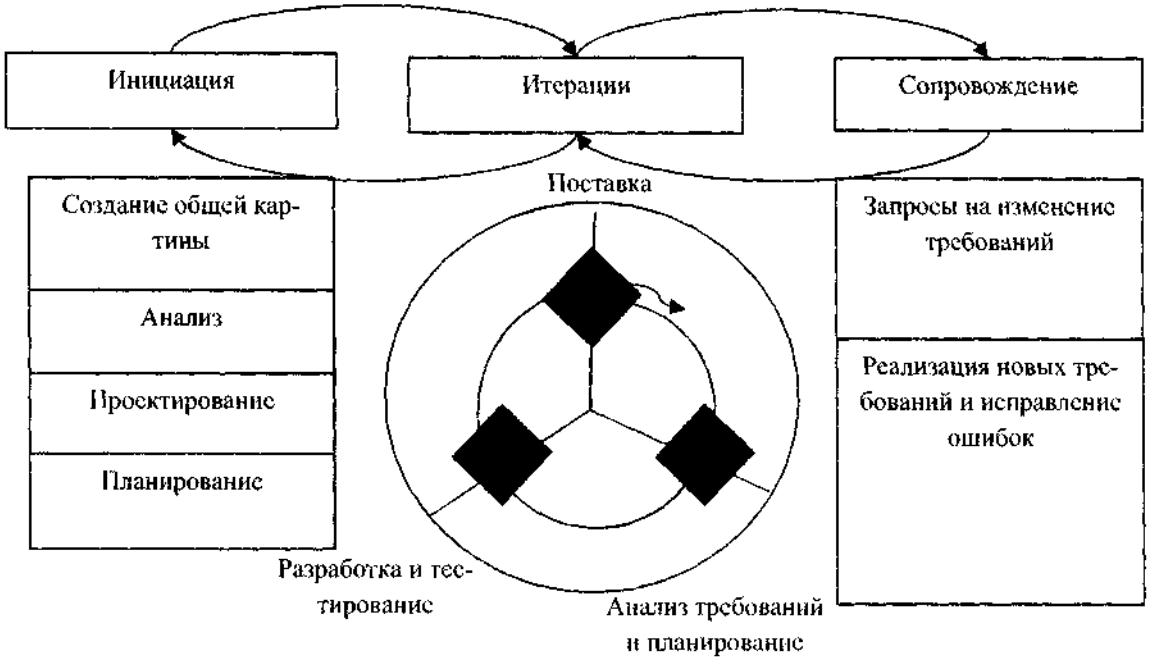

Рис. 1. Взаимосвязь этапов разработки ПС

Стадия итераций – это непосредственно совокупность актов *реализации и поставки* работающего ПС. Длительность итерации не должна превышать одной недели, рекомендуемой же практикой являются ежедневные сборки. Итерация начинается с того, что происходит анализ требований, которые должны быть реализованы на текущем витке разработки, осуществляется планирование процесса разработки. Далее выполняется реализация и тестирование ПС. После этого оттестированное ПС передается пользователю. Такой итеративный подход позволяет быстро получать отзывы пользователя о ПС. Как правило, этот отзыв приходит в виде новых требований, отказа от старых или модификации существующих требований.

На рисунке 2 поясняется итеративность предлагаемого подхода.

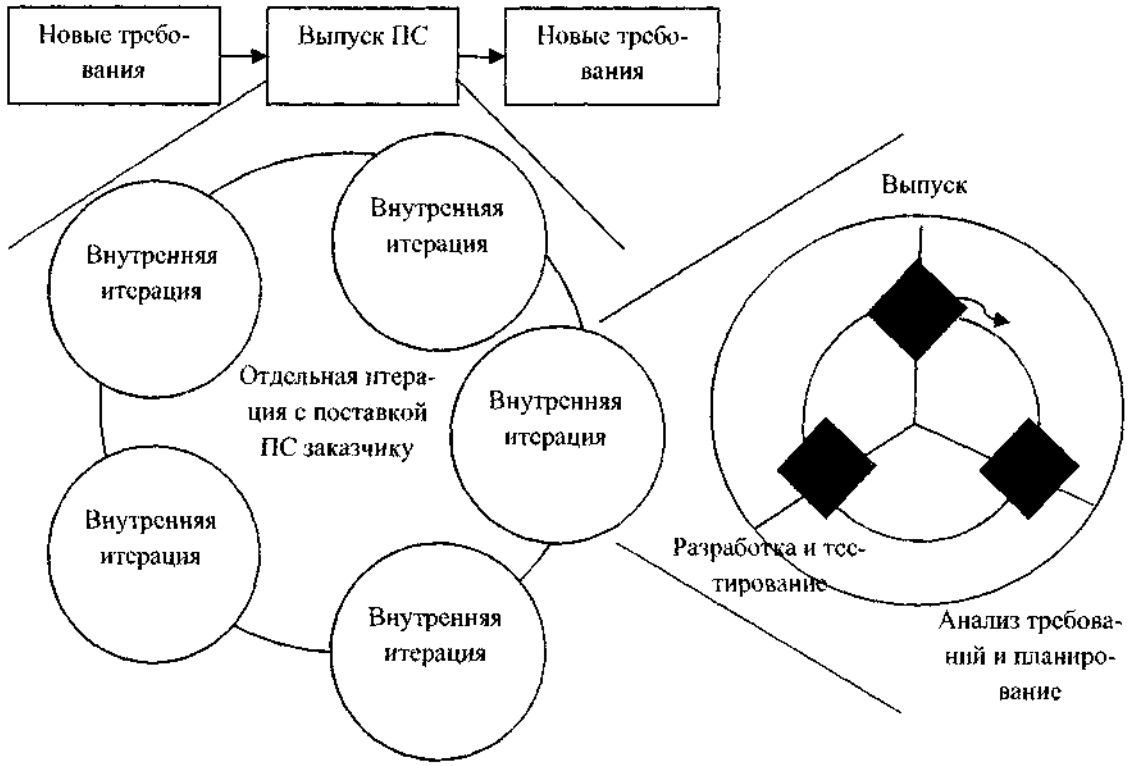

Рис. 2. Итеративность процесса разработки ПС

Для эффективной реализации качественного ПС предлагается использовать два вида итераций:

- итерации с поставкой ПС заказчику;
- внутренние итерации, каждая из которых предназначена главным образом для команды инженеров по качеству; внутренние итерации показывают прогресс разработки ПС и удостоверяют отсутствие ошибок в поставляемой заказчику версии ПС.

Сопровождение – этап, начало которого знаменует завершение основных активностей, касающихся разработки ПС. На этом этапе выполняется исправление ошибок, которые не были найдены или устранены при непосредственной реализации ПС.

Важно понимать наличие связей между этапами разработки. Очевидно, как бы хорошо ни был налажен итерационный процесс разработки, определяющий цикл «анализ требований и планирование – разработка и тестирование – поставка», нельзя не принимать во внимание то, что при появлении принципиально новых требований придется вернуться на этап инициации, и создать новую картину решаемой задачи, провести анализ предметной области, затрагивающий новые, теперь уже необходимые, нюансы. Также и на этапе сопровождения, изменение требований ведут к возврату на итерационный процесс разработки. Более того, изменения требований на этапе сопровождения могут быть настолько существенными, что потребуется выполнить возврат на этап инициации, и полностью повторить все работы данного этапа.

Заключение

Подводя итог вышесказанному, в данной работе предлагается адаптировать гибкие методы разработки ПС путем использования подмножества из трех методов: метода гибкого моделирования, метода экстремального программирования и метода разработки, ведомой функциями. Гибкое моделирование позволяет эффективно создавать архитектуру ПС в условиях изменяющихся требований. Для качественной реализации требований используются приемы и подходы экстремального программирования. Разработка, ведомая функциями, позволяет управлять созданием ПС. Предлагается использовать три этапа при создании ПС: инициация, стадия итераций и сопровождение. Акцент делается на итеративной разработке, причем выделяются итерации с поставкой ПС заказчику и внутренние итерации.

Литература

1. *Hunt, J.* Agile Software Construction / J. Hunt. – Springer-Verlag London Limited, 2006. – 254 p.
2. *Carmichael, A.* Better Software Faster / A. Carmichael, D. Haywood // Prentice-Hall, 2002. – 384 p.
3. *Beck, K.* Манифест гибкой разработки [Электронный ресурс] / K. Beck [и др.] // Agilemanifesto – 2001. – Режим доступа: http://agilemanifesto.org/. – Дата доступа: 15.03.2009.
4. *Ambler, S., W.* Agile Modeling: Effective Practices for Extreme Programming and the Unified Process / Scott W. Ambler, Ron Jeffries / Wiley. – 2002. – 224 p.
5. *Beck, K.* Extreme Programming Explained: Embrace Change / K. Beck, Ch. Andres. – 2 ed., Addison Wesley, 2004. – 224 p.
6. *Burke, E. M.* Java Extreme Programming Cookbook / E. M. Burke, B. M. Coyner. – O'Reilly Media, Inc., 2003. – 352 p.
7. *Palmer, S. R.* A Practical Guide to Feature-Driven Development / S. R. Palmer, J. M. Felsing. – Prentice Hall, 2002. – 304 p.

Неборский Сергей Николаевич, аспирант Белорусского государственного университета информатики и радиоэлектроники, магистр технических наук, sergei.neborski@gmail.com