4A

УДК 004.05

Абасова Судаба Эйбала гызы, Абдуллаев Сайяр Габиб оглу
Институт информационных технологий Национальной академии наук Азербайджана,
г. Баку, Азербайджан
depart5@iit.ab.az

Об одном подходе к разработке качественного программного обеспечения

В статье рассмотрены пути и модели разработки качественного программного обеспечения. Представлены некоторые технологии в рамках реализации формальных методов, модели совершенствования программных процессов и методология разработки программного обеспечения. Также дана информация о разработанной корпоративной информационной системе «Бухгалтерия» с помощью концепций CBSE (компонентная разработка программного обеспечения) и системе визуального программирования RAD (быстрая разработка приложений).

Введение

Программное обеспечение разрабатывают уже больше пятидесяти лет, но до сих пор программы, изобилующие ошибками, остаются нормой, а качественные решения – редчайшим исключением. Однако до сих пор не существует общих технологий, которые позволили бы всем разработчикам писать надежное программное обеспечение с приемлемыми затратами и в разумное время.

Индустрия программного обеспечения постоянно пытается решить вопрос качества, но насколько значимы ее успехи, на данный момент сказать довольно сложно.

Таким образом, уровень ошибок последние двадцать лет практически не меняется, несмотря на объектно-ориентированную технологию, автоматические отладчики, более качественные средства тестирования и более строгий контроль типов в таких языках, как Java и Ada.

Проблема повышения качества программного обеспечения в целом и качества тестирования в частности привлекает все большее внимание; в университетах вводят специальные дисциплины по тестированию и обеспечению качества, готовят узких специалистов по тестированию и инженеров по обеспечению качества. Однако по-прежнему ошибки стоят очень дорого, только в США на это тратят от 20 до 60 млрд долларов ежегодно. При этом примерно 60 % убытков ложится на плечи конечных пользователей. Складывается ситуация, при которой потребители вынуждены покупать заведомо бракованный товар.

Вместе с тем ситуация не безнадежна. Исследование, проведенное Национальным институтом стандартов и технологии США, показало, что размер убытков, связанных со сбоями в программном обеспечении, можно уменьшить примерно на треть, если вложить дополнительные усилия в инфраструктуру тестирования, в частности в разработку инструментов тестирования.

Чем позже будут обнаружены ошибки и уязвимые места программы, тем дороже обойдется их исправление, поэтому наиболее оптимальным, как нам представляется, является решение проблемы уже в процессе разработки.

Некоторые программисты сейчас отказываются от традиционных методов разработки программного обеспечения (как одномоментных, так и поэтапных) в пользу методов быстрого и экстремального программирования. В своем крайнем проявлении быстрая разработка RAD (Rapid Application Development – быстрая разработка приложений) – это абсолютно неструктурированный, хаотический процесс, который использует специализированные методы и в котором пропускаются многие этапы планирования и проектирования.

Постановка задачи

В данной статье предлагаются следующие методы решения:

– применить спиральную модель жизненного цикла программного обеспечения;
– применить концепцию компонентной разработки программного обеспечения (CBSE);
– применить визуальное (объектно-ориентированное) программирование – быструю разработку приложений (RAD), а при конструировании интерфейса приложения принцип WYSIWYG (What You See Is What You Get – что видите, то и получите) для разработки качественного программного обеспечения.

Борьба за качество программ может вестись двумя путями. Первый путь простой: собрать команду квалифицированных программистов с опытом участия в аналогичных проектах, дать им правильно поставленную задачу, хорошие инструменты, создать надлежащие условия работы. С большой вероятностью можно ожидать, что им удастся разработать программную систему с хорошим качеством.

Второй путь не так прост, но позволяет получать качественные программные продукты и тогда, когда перечисленные условия соблюсти не удается, – не хватает хороших программистов, четкости в постановке задачи и т.д. Этот путь предписывает стандартизировать процессы разработки: ввести единообразные требования к этапам работ, документации, организовать регулярные совещания, проводить инспекцию кода и пр. Одним из первых продвижений на этом поприще стало введение понятия жизненного цикла программной системы, четко определявшего необходимость рассмотрения многих задач, без решения которых нельзя рассчитывать на успех программного проекта.

В простейшем варианте набор этапов жизненного цикла таков:

1) анализ требований;
2) проектирование (предварительное и детальное);
3) кодирование и отладка («программирование»);
4) тестирование;
5) эксплуатация и сопровождение.

Следует отметить, что наиболее дорогие ошибки совершаются на первых фазах жизненного цикла – это ошибки в определении требований, выборе архитектуры, высокоуровневом проектировании. Поэтому надо концентрироваться на поиске ошибок на всех фазах, включая самые ранние, не дожидаясь, пока они обнаружатся при тестировании уже готовой реализации.

Методы решения

Стандартизированная схема жизненного цикла с четкой регламентацией необходимых работ и перечнем соответствующей документации легла в основу так называемой «**водопадной**» или «**каскадной**» модели (рис. 1).

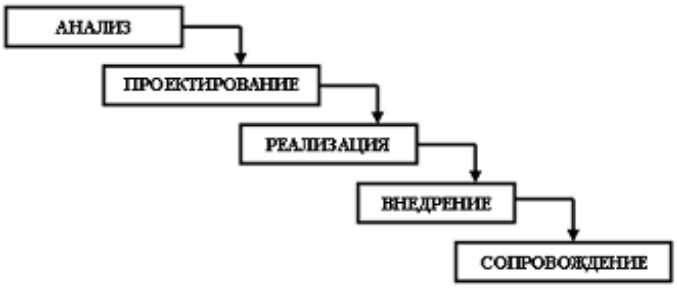


Рисунок 1 – «Водопадная» или «каскадная» модель процесса разработки

«Водопадная» модель подразумевает жесткое разбиение процесса разработки программного обеспечения на этапы, причем переход с одного этапа на другой осуществляется только после того, как будут полностью завершены работы на предыдущем этапе. Каждый этап завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой. «Водопадная» модель требовала точно и полно сформулировать все требования; изменение требований было возможно только после завершения всех работ. «Водопадная» модель не давала ответ на вопрос, что делать, когда требования меняются или меняется понимание этих требований непосредственно во время разработки.

Потребовалось изменить процесс разработки так, чтобы гарантировать внесение необходимых исправлений уже после завершения какого-либо этапа. Так появилась модель «**водоворота**» или «**возвратная**» модель (рис. 2).

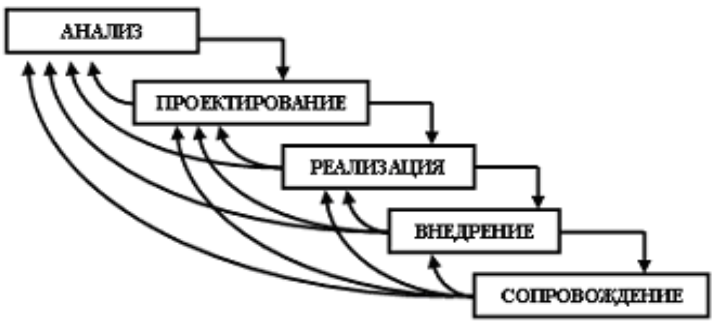


Рисунок 2 – Модель «водоворота» или «возвратная» модель процесса разработки

Здесь мы видим, что недостатки проектирования и программирования могут быть устранены позже путем частичного возврата продукта на предыдущую стадию. Само собой разумеется, чем ниже уровень, на котором обнаружена ошибка, тем дороже обходится ее исправление. Приводятся такие цифры, как десятикратное возрастание затрат на переделку с каждым следующим этапом.

В такой ситуации огромное значение приобретает этап формулирования требований, составления спецификаций и создания плана системы. Программные архитекторы несут чуть ли не личную ответственность за все последующие изменения проектных решений. Объем документации исчисляется тысячами страниц, а количество заседаний, созванных для утверждения и отнимающих рабочее время многих людей, просто огромно. Да иначе и быть не может, так как попытки предусмотреть весь

цикл разработки и некоторое время эксплуатации продукта влекут фантастические затраты людских, материальных и временных ресурсов. Эта проблема была решена в следующей модели, называемой «**спиральной**» (рис. 3).

Рисунок 3 – «Спиральная» модель процесса разработки

Эта методология является наиболее распространенной в текущее время. Самыми известными ее вариантами являются RUP (Rational Unified Process) от фирмы Rational и MSF (Microsoft Solution Framework). Теперь создание системы предполагается проводить итерационно, двигаясь по спирали и проходя через одни и те же стадии, на каждом витке уточняя характеристики будущего продукта. Казалось бы, теперь все хорошо: и планируем мы только то, что можем предвидеть, и разрабатываем то, что запланировано, и пользователи начинают знакомиться с продуктом заранее, имея возможность внести необходимые коррективы.

Только для этого нужны очень большие средства. Действительно, если раньше можно было создавать и распускать группы специалистов по мере необходимости, то теперь все они должны постоянно участвовать в проекте: архитекторы, программисты, тестировщики, инструкторы и т.д. Более того, усилия различных групп должны быть синхронизированы, чтобы своевременно отражать проектные решения и вносить необходимые изменения.

За прошедшие годы разработчики программного обеспечения сформулировали множество принципов совершенствования методов программирования, часть которых уже подтвердила свою эффективность в практических проектах. К ним относятся методологии и среды разработки программного обеспечения, структурное и объектно-ориентированное программирование, модели совершенствования программных процессов (Capability Maturity Model, CMM), средства автоматизации разработки программного обеспечения (Computer Aided Software Engineering, CASE – разработка ПО с помощью компьютера) и языки четвертого поколения. Тем не менее пока остаются нерешенными некоторые проблемы.

1990-е годы ознаменовали первую реальную попытку превратить разработку программного обеспечения в инженерную дисциплину с помощью концепций CBSE (component-based software engineering – компонентная разработка программного обеспечения) и COTS (commercial off-the-shelf – готовые коммерчески доступные компоненты). Идея состоит в создании небольших, высококачественных модулей и последующем их объединении. Все согласны, что языки четвертого поколения (4GL – Visual Basic, Delphi, Си, Си++, Oracle) позволяют увеличить производительность программистов и сократить число потенциальных ошибок.

Можно сказать, что Delphi относится к системам визуального программирования, которые называются также системами RAD (Rapid Application Development – быстрая разработка приложений). Разработка приложения в Delphi включает два взаимосвязанных этапа:

– создание интерфейса приложения;

– определение функциональности приложения.

Интерфейс приложения определяет способ взаимодействия пользователя и приложения, то есть внешний вид формы (форм) при выполнении, и то, каким образом пользователь управляет приложением. Интерфейс конструируется путем размещения на форме компонентов, называемых интерфейсами или управляющими компонентами. Создается интерфейс приложения с помощью конструктора формы.

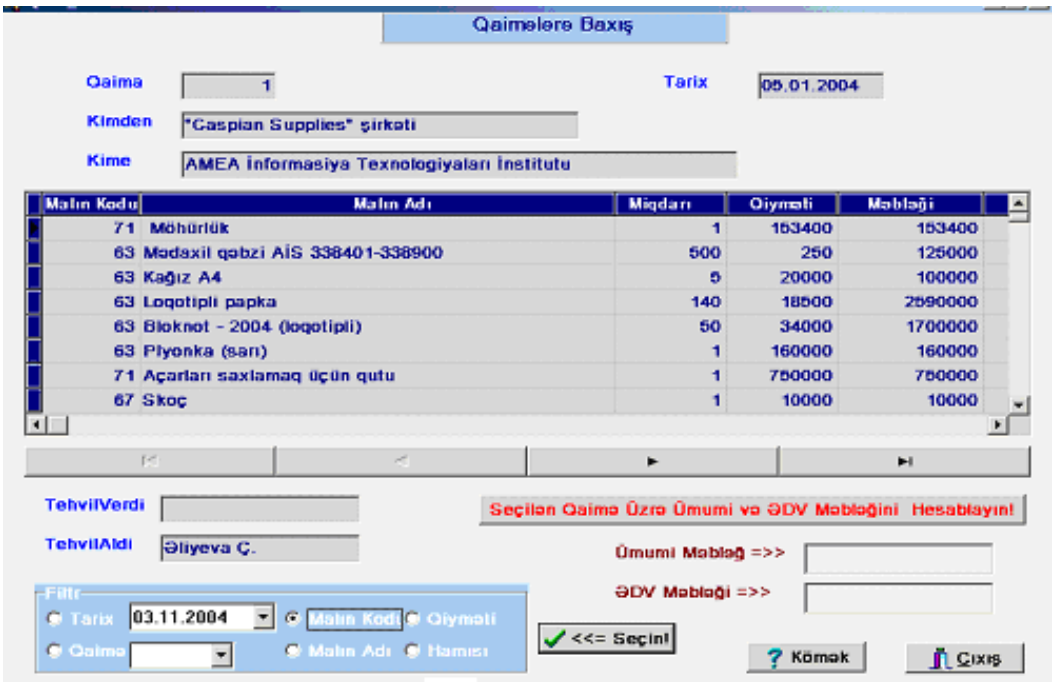
Функциональность приложения определяется процедурами, которые выполняются при возникновении определенных событий, например, происходящих при действиях пользователя с управляющими компонентами формы.

Таким образом, в процессе разработки приложения на форму помещаются компоненты и для них устанавливаются свойства и создаются обработчики событий.

Интерфейс приложения составляют компоненты, которые разработчик выбирает из палитры компонентов и размещает на форме, то есть компоненты являются своего рода строительными блоками. При конструировании интерфейса приложения действует принцип WYSIWYG (What You See Is What You Get – что видите, то и получите), и разработчик при создании приложения видит форму почти такой же, как и при ее выполнении.

В качестве примера отметим, что в разработанной нами корпоративной информационной системе «Бухгалтерия» (задача «Учет движения материалов») на языке программирования Delphi 6 (4-го поколения) использованы концепция CBSE (компонентная разработка программного обеспечения) и визуальное программирование RAD (быстрая разработка приложений) (табл. 1).

Таблица 1



Выводы

В заключение можно сказать, что в статье показаны пути решения проблем, связанных с ошибками, возникшими в процессе разработки программного обеспечения. Кроме того, на различных этапах жизненного цикла процесс разработки программного обеспечения, применения новых методологий, моделей, средств и методов дает возможность с приемлемыми затратами и в разумное время разработать более качественное программное обеспечение.

Литература

1. Джеймс Уайттеккер, Джеффри Воас. 50 лет программирования: основные принципы качества // Открытые системы. – 2003. – № 3.
2. Александр Петренко, Елена Бритвина, Сергей Грошев, Александр Монахов, Ольга Петренко. Тестирование на основе моделей // Открытые системы. – 2003. – № 9.
3. Александр Родыгин. Процесс разработки или разрабатываем процесс. – Режим доступа: http://www.i2r.ru/static/371/out-8542.shtml.
4. Владимир Гофман, Анатолий Хомоненко. Delphi 6. – 2001.
5. Abasova S.E., Abdullayev S.H. AMEA Известия. – 2005. – T. XXV, № 3.

Абасова Судаба Ейбала гизи, Абдулаєв Сайяр Габіб оглу

Про один підхід до розробки якісного програмного забезпечення

Розглянуті шляхи та моделі розробки якісного програмного забезпечення. Наведені деякі технології у рамках реалізації формальних методів, моделі вдосконалення програмних процесів і методологія розробки програмного забезпечення. Також дана інформація щодо розробленої корпоративної інформаційної системи «Бухгалтерія» при допомозі концепцій CBSE (компонентна розробка програмного забезпечення) та систем візуального програмування RAD (швидка розробка додатків).

S.A. Abasova, S.G. Abdullayev

On One Approach to the Qualitative Software

There are presented some technologies within the limits of realization of formal methods, the models of perfection of program processes and methodology of software development. Also is given the information about developed corporate information system «Accounting» by means of CBSE concepts and the system of visual programming RAD is developed.

Статья поступила в редакцию 22.05.2008.