УДК 681.321

МЕТОДЫ ОБЕСПЕЧЕНИЯ КАЧЕСТВА ПРИ ПРОЕКТИРОВАНИИ СЛОЖНЫХ ПРОГРАММНЫХ СИСТЕМ

А. А. Карпунин, Ю. М. Ганев, М. М. Чернов

Введение

Согласно ISO 8402 качество программного обеспечения – совокупность характеристик ПО, относящихся к его способности удовлетворять установленные и предполагаемые потребности.

Первой задачей каждого разработчика является определение целей и потребностей, представленных в виде требований к программному продукту. Они включают в себя как требования к функциональности (функциональные требования), так и требования к качеству разрабатываемой системы (нефункциональные требования). Ошибка в толковании требований на ранних этапах может привести к значительным затратам при доработке готового продукта, несоответствию результатов работы ожиданиям заказчика. Таким образом, в самом начале процесса разработки программной системы следует с полной ответственностью подойти к выявлению и документированию требований, в число которых входят заданные характеристики качества [1–4].

Анализ характеристики качества программных систем

Качество программного обеспечения состоит из трех аспектов:
– качество процесса;
– качество продукта;
– качество сопровождения (рис. 1).

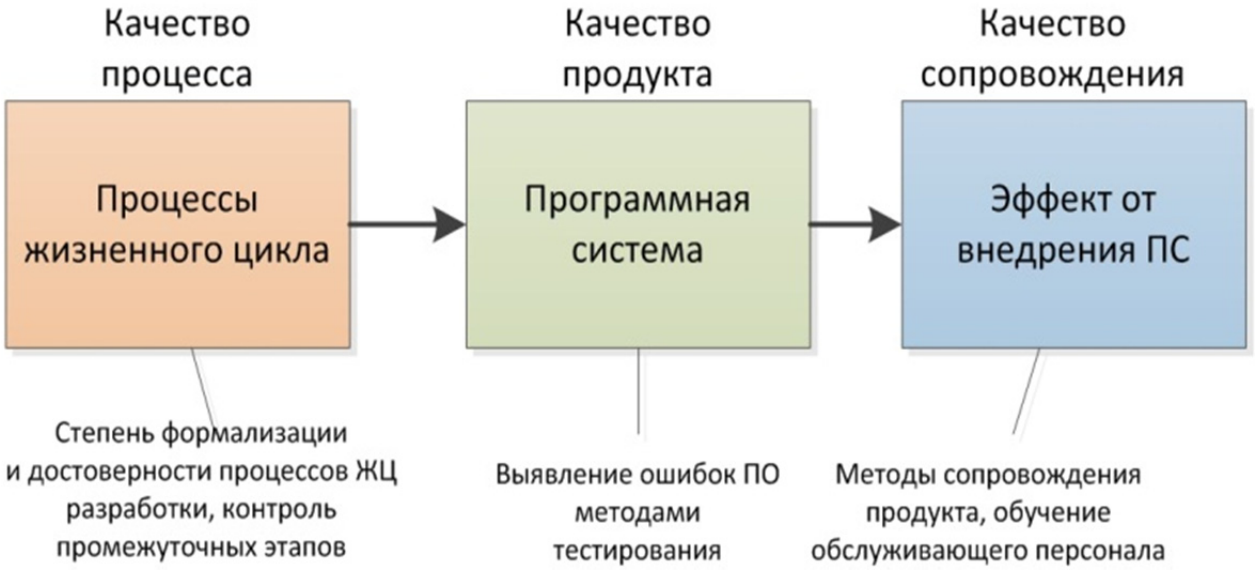

Рис. 1. Аспекты качества программных систем

Далее определяются характеристики качества, отражающие различные точки зрения конечного пользователя на качество (рис. 2).

Рис. 2. Обобщенная модель качества программных систем

Функциональность – это способность ПО решать задачи, поставленные пользователем. Это определенный набор характеристик, которые одновременно соответствуют потребностям

пользователей и являются определяющим показателем для работы ПО, т.е. это исправность и точность работы ПО.

Надежность – способность ПО к бесперебойной работе в течение определенного промежутка времени. Она характеризуется отказоустойчивостью, способностью к самостоятельному восстановлению, а также целостностью и завершенностью всей системы.

Удобство использования – это способность ПО быть интуитивно понятным и привлекательным пользователю.

Эффективность – это возможность ПО обеспечить наиболее быстрое достижение целей и решение задач, поставленных пользователем.

Удобство сопровождения – возможность для администрирующего персонала оперативно и быстро реагировать на изменение системы, быстрая адаптация системы под меняющиеся задачи и требования пользователей.

Портативность – способность ПО не быть привязанным к техническим характеристикам устройства, на котором оно установлено. Возможность быстрого переноса из одной среды в другую [6–8].

Основными факторами неудовлетворительного качества являются:

1) функциональные ошибки – несоответствия требованиям, техническому заданию;

2) нефункциональные ошибки – нарушение правил языка программирования, использования библиотечных функций и сторонних компонентов и т.п.;

3) ошибки при контроле – неправильно выполненная оценка трудозатрат, неверно составленный план работ;

4) недостаточная прозрачность процесса разработки – в каждый момент времени сложно оценить состояние проекта и процент его завершения;

5) отсутствие гарантий качества – из-за вероятности возникновения ошибок на всем протяжении жизненного цикла программной системы.

Методы улучшения качества кода

Общепринятые концепции совершенствования качества, такие как предотвращение и ранняя диагностика ошибок, постоянное совершенствование (continuous improvement) и внимание к требованиям заказчика (customer focus), применимы и к программной инженерии [5–12]. Они основываются на работах экспертов по качеству, пришедших к мнению, что качество продукта напрямую связано с качеством используемых для его создания процессов.

Наиболее известными подходами к улучшению качества являются TQM (Total Quality Management), PDCA (Plan, Do, Check, Act), итеративная разработка [13–19]. Немаловажным условием успешного выполнения процессов, оценки продуктов и получения всех необходимых данных является поддержка со стороны менеджмента [12].

SQM (Software Quality Management) – управление качеством программного обеспечения, SQM применяется ко всем процессам и их аспектам, сопровождающим создание программного обеспечения, а именно: к требованиям, к владельцам процессов, к изменениям процессов, к результатам этих изменений.

Процессы управления качеством ПО состоят из большого числа пунктов. Главной их целью является преждевременное нахождение ошибок и дефектов создаваемой системы. В одних случаях эти процессы помогают найти сам дефект напрямую, в других – всего лишь указать направление дальнейших исследований.

Процессы планирования качества отличаются от процессов, направленных на оценку характеристик качества, а не на настоящую реализацию. Процессы управления качеством должны быть направлены на удовлетворение потребностей пользователей и заинтересованных в этом проекте лиц, гарантировать исправную работу ПО и иметь ценность для заказчика [8–11].

SQM может использоваться для оценки и конечных, и промежуточных продуктов.

Некоторые из специализированных процессов SQM определены в стандарте 12207:

– процесс обеспечения качества (*quality assurance process*);

– процесс верификации (*verification process*);

– процесс аттестации (*validation process*);

– процесс совместного анализа (*joint review process*);

– процесс аудита (*audit process*).

Эти процессы направляют к получению качества, помогают в поиске ошибок.

Процессы SQM улучшают качество программного обеспечения в проекте. Они дают руководителям основную информацию о всех проблемах и параметрах данного продукта.

Процессы SQM состоят из методов и техник, которые предназначены для оценки того, как реализуется создание ПО и насколько соответствуют промежуточные и конечные версии требованиям к продукту. Благодаря этим процессам создаются отчеты для руководителей, которые могут своевременно предпринять корректирующие действия. Данные в отчетах должны быть достоверными.

Рефакторинг – это процесс изменения существующего кода и структуры программы, не изменяющий при этом функционал и поведение системы [4]. Целью рефакторинга является перестройка и улучшение существующего кода, это улучшение проявляется в логичности кода, в более четкой и точной структуре. Благодаря рефакторингу улучшается качество системы. Он состоит из ряда действий, которые сами по себе являются небольшими, но при этом абсолютно эквивалентны работе системы и ее функционалу.

В экстремальном программировании и других гибких методологиях рефакторинг является неотъемлемой частью цикла разработки ПО [12]: разработчики попеременно то создают новые тесты и функциональность, то выполняют рефакторинг кода для улучшения его логичности и прозрачности. Иногда под рефакторингом неправильно подразумевают коррекцию кода с заранее оговоренными правилами отступа, перевода строк, внесения комментариев и прочими визуально значимыми изменениями, которые никак не отражаются на процессе компиляции, с целью обеспечения лучшей читаемости кода.

Рефакторинг изначально не предназначен для исправления ошибок и добавления новой функциональности. Но это помогает избежать ошибок и облегчить добавление функциональности. Он выполняется для улучшения понятности кода или изменения его структуры, для удаления «мертвого кода» – все это для того, чтобы в будущем код было легче поддерживать и развивать. В частности, добавление в программу нового поведения может оказаться сложным с существующей структурой – в этом случае разработчик может выполнить необходимый рефакторинг, а уже затем добавить новую функциональность.

Рефакторинг нужно применять постоянно, так как разрабатываемый код может быть очень сложным и объемным. Основными причинами могут быть:

– обновление кода, встраивание нового функционала в код;

– разрешение скрытых ошибок;

– сложная логика системы при командной работе над кодом.

Реинжиниринг – это процесс создания нового функционала программного обеспечения, а также устранения ошибок с помощью революционного изменения кода, используемого в существующем и работающем ПО [14–20]. Выражаясь менее формально, реинжиниринг является изменением системы программного обеспечения после проведения обратного инжиниринга.

Считается, что намного легче создать новый программный продукт, чем переделывать старый. Это связано со следующими проблемами:

– неопытному программисту сложно разобраться в чужом коде, особенно, если его структура не логична и не прозрачна для понимания;

– реинжиниринг обычно дороже, чем разработка нового ПО. Это связано с тем, что при создании нового ПО не учитывается его связь и совместимость с предыдущими версиями;

– неопытные программисты обычно проводят реинжиниринг некачественно, так как не обладают достаточной квалификацией. Поэтому для этого процесса необходимы программисты более высокой квалификации, соответственно, их время работы оценивается дороже.

Заключение

Анализ методов улучшения качества программного обеспечения привел к тому, что наиболее перспективными методами улучшения кода программного обеспечения являются тестирование, рефакторинг и реинжиниринг. Все эти методы являются относительно новыми и постоянно развивающимися. Самым простым решением являются рефакторинг кода и его тестирование. И тот, и другой методы являются не такими трудозатратными, как реинжиниринг, который требует как высокой квалификации программиста, так и большого количества времени.

В заключение можно порекомендовать для начала использовать тестирование, так как это логичный и самый очевидный метод улучшения качества программного обеспечения. Тестирование является неоднозначным процессом, так как требует творческого подхода и достаточно большой фантазии. Чем большая часть кода покрыта тестами, тем оно эффективней, тем более качественный код вы получите на выходе.

В дальнейшем можно порекомендовать постепенно вводить и выделять время на рефакторинг кода. Этот метод позволит качественно улучшить существующую архитектуру, сделать ее логичной и прозрачной. Что впоследствии скажется на добавлении нового функционала в систему и взаимодействии этой системы с предыдущими ее версиями.

Список литературы

1. Конструкторско-технологическое проектирование электронных средств / К. И. Билибин, А. И. Власов, Л. В. Журавлева и др. ; под общ. ред. В. А. Шахнова. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2005. – 568 с. – (Информатика в техническом университете ; изд. второе).
2. Хэнсен, Б. Л. Контроль качества. Теория и применение : пер. с англ. / Б. Л. Хэнсен. – М. : Прогресс, 1968.
3. Статистические методы контроля качества продукции : пер. с англ. / Л. Ноулер и др. – М. : Изд-во стандартов, 1989. – 96 с.
4. Рефакторинг. Улучшение существующего кода / Мартин Фаулер, Кент Бек, Джон Брант, Уильям Апдайк, Дон Робертс, Эрих Гамма. – СПб. : Символ-Плюс, 2008. – 432 с.
5. Глудкин, О. П. Всеобщее управление качеством : учеб. для вузов / О. П. Глудкин. – М. : Радио и связь, 1999.
6. Власов, А. И. Визуальные модели управления качеством на предприятиях электроники / А. И. Власов, А. М. Иванов // Наука и образование. – 2011. – № 11. – С. 34–36.
7. Современные методы и средства обеспечения качества в условиях комплексной автоматизации / В. Г. Дудко, К. Д. Верейнов, А. И. Власов, А. Г. Тимошкин // Вопросы радиоэлектроники. Сер. АСУПР. – 1996. – № 2. – С. 54–72.
8. Тимошкин, А. Г. О стратегии и тактике маркетинговой политики многопрофильной компьютерной фирмы / А. Г. Тимошкин, А. И. Власов // Приборы и системы управления. – 1996. – № 9. – С. 59–61.
9. Маркелов, В. В. Системный анализ процесса управления качеством изделий электронной техники / В. В. Маркелов, А. И. Власов, Э. Н. Камышная // Надежность и качество сложных систем. – 2014. – № 1. – С. 35–43.
10. Власов, А. И. Системный анализ технологических процессов производства сложных технических систем с использованием визуальных моделей / А. И. Власов // Международный научно-исследовательский журнал. – 2013. – № 10, ч. 2. – С. 17–26.
11. Власов, А. И. Методы генерационного визуального синтеза технических решений в области микронаносистем / А. И. Власов, Л. В. Журавлева, Г. Г. Тимофеев // Научное обозрение. – 2013. – № 1. – С. 107–111.
12. Власов, А. И. Пространственная модель оценки эволюции методов визуального проектирования сложных систем / А. И. Власов // Датчики и системы. – 2013. – № 9. – С. 10–28.
13. Власов, А. И. Визуализация творческих стратегий с использованием ментальных карт / А. И. Власов, Л. В. Журавлева // Прикаспийский журнал: управление и высокие технологии. – 2013. – № 1. – С. 133–140.
14. John Bergey, William Hefley, Walter Lamia, Dennis Smith A Reengineering Process Framework, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1995. – P. 479–481.
15. Власов, А. И. Информационно-управляющие системы для производителей электроники / А. И. Власов, А. Е. Михненко // Производство электроники: технологии, оборудование материалы. – 2006. – № 3. – С. 15–21.
16. Власов, А. И. Принципы построения и развертывания информационной системы предприятия электронной отрасли / А. И. Власов, А. Е. Михненко // Производство электроники: технологии, оборудование, материалы. – 2006. – № 4. – С. 5–12.
17. Баранов, Н. А. Управление состоянием готовности системы безопасности к отражению угрозы / Н. А. Баранов, Н. А. Северцев // Труды Международного симпозиума Надежность и качество. – 2012. – Т. 1. – С. 8–10.
18. Дедков, В. К. Компьютерное моделирование характеристик надежности нестареющих восстанавливаемых объектов / В. К. Дедков, Н. А. Северцев // Труды Международного симпозиума Надежность и качество. – 2010. – Т. I. – С. 368–370.

19. Адамова, А. А. Моделирование бизнес-процесса корпорации на примере электронных средств / А. А. Адамова, А. В. Черняев // Проектирование и технология электронных средств. – 2002. – № 5. – С. 86–92.
20. Автоматизированный реинжиниринг программ : сб. ст. / под ред. А. Н.Терехова, А. А. Терехова. – СПб. : Изд-во Санкт-Петербургского университета, 2000. – 256 с.

Карпунин Алексей Александрович
ассистент,
кафедра проектирования и технологии производства электронной аппаратуры,
Московский государственный
технический университет им. Н. Э. Баумана
(105005, Россия, г. Москва,
2-я Бауманская ул., д. 5, стр. 1)
8-499-263-65-53
E-mail: AlexK811@yandex.ru

Ганев Юрий Михайлович
лаборант-исследователь,
кафедра проектирования и технологии производства электронной аппаратуры,
Московский государственный
технический университет им. Н. Э. Баумана
(105005, Россия, г. Москва,
2-я Бауманская ул., д. 5, стр. 1)
8-499-263-65-53
E-mail: yury.ganev@gmail.com

Чернов Максим Михайлович
аспирант,
кафедра проектирования и технологии производства электронной аппаратуры,
Московский государственный
технический университет им. Н. Э. Баумана
(105005, Россия, г. Москва,
2-я Бауманская ул., д. 5, стр. 1)
Телефон: 8-499-263-65-53

Аннотация. В работе проанализированы способы обеспечения надлежащего качества сложных программных систем. Основное внимание уделено вопросам качества программного обеспечения, структурирования программного кода и архитектуры программного обеспечения. В настоящее время проблемы, связанные с качеством программного обеспечения, в сущности связанные с качеством написанного кода, становятся весьма наболевшим и актуальным вопросом в мире программного обеспечения и IT-сфере. Риски, связанные с некачественным ПО, постоянно растут, как для пользователей, так и для создателей. В статье кратко исследованы вопросы по структуре качества программного обеспечения, исследованы причины появления проблемного ПО. Также рассмотрены вопросы тестирования программного обеспечения. Вопросы тестирования и создания специальных тестов являются весьма важными для качества программного обеспечения. Некачественное тестирование ведет к появлению серь-

Karpunin Aleksey Aleksandrovich
assistant,
sub-department of design and production technology of the electronic equipment,
Moscow State Technical University
named after N. E. Bauman
(105005, 2-ya Baumanskaya street, apartment 5, page 1, Moscow, Russia)

Ganev Yuriy Mikhaylovich
laboratory assistant-researcher,
sub-department of design and production technology of the electronic equipment,
Moscow State Technical University
named after N. E. Bauman
(105005, 2-ya Baumanskaya street, apartment 5, page 1, Moscow, Russia)

Chernov Maksim Mikhaylovich
postgraduate student,
sub-department of design and production technology of the electronic equipment,
Moscow State Technical University
named after N. E. Bauman
(105005, 2-ya Baumanskaya street, apartment 5, page 1, Moscow, Russia)

Abstract. This paper analyzes the ways of ensuring proper quality of complex software systems. The main attention is paid to the quality of the software, structuring code and software architecture. At the moment the problems associated with the quality of software, in essence connected with the quality of the written code, become very sore and relevant issue in the world of software and IT-sphere. Improperly written software brings future difficulties for the administration, updating and implementation of the new versions. Risks associated with low-quality software is constantly growing, both for users and for the creators. The article briefly researched questions on the structure of software quality, investigated the causes of the problem software. Software testing is also considered. Questions of testing and creation of special tests are very important for the quality of the software. Poor testing leads to serious errors in the subsequent release to the market and disruptions in the software. Testing is one of the methods to improve the code, and in general the quality of the entire soft-

езных ошибок в последующем выпуске на рынок и работе программного обеспечения. Приведены различные методы по улучшению написанного кода, яркими их представителями являются рефакторинг и реинжиниринг кода. Эти методы являются наиболее перспективными и актуальными в настоящее время. В заключении даны общие характеристики этих методов, а также рекомендации по их использованию.

Ключевые слова: качество, программное обеспечение, жизненный цикл, функциональность, эффективность, портативность.

ware. This article shows various methods for improving the written code, with bright their representatives are refactoring and reengineering of code. These methods are the most promising and relevant today. In conclusion, the general characteristics of these methods, as well as recommendations for their use, are given.

Key words: quality, software lifecycle functionality, efficiency, portability.

УДК 681.321

Карпунин, А. А.

Методы обеспечения качества при проектировании сложных программных систем / А. А. Карпунин, Ю. М. Ганев , М. М. Чернов // Надежность и качество сложных систем. – 2015. – № 2 (10). – С. 78–84.