УДК 004.054
doi:10.18720/SPBPU/2/id20-226

Калугян Каринэ Хачересовна[1],
канд. экон. наук, доцент, доцент кафедры Информационных систем и прикладной информатики;
Щербакова Капитолина Николаевна[2],
зав. сектором контроля качества;
Глущенко Марина Владимировна[3],
магистрант кафедры
Информационных систем и прикладной информатики

МОДЕЛИРОВАНИЕ ПРОЦЕССОВ КОНТРОЛЯ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

[1, 3] Ростовский государственный экономический университет (РИНХ), Ростов-на-Дону, Россия,
[1] kalugyan@yandex.ru, [3] mary_kush@mail.ru
[2] НИИ «Спецвузавтоматика», Ростов-на-Дону, Россия,
kapitolina-orlova@yandex.ru

Аннотация. Рассматриваются вопросы осуществления контроля качества разрабатываемого программного обеспечения путем проведения тестирования. Для визуального моделирования процессов тестирования используется язык UML.

Ключевые слова: программное обеспечение, контроль качества, тестирование, моделирование.

Karine Kh. Kalugyan[1],
Associate Professor, The Department of Information Systems and Applied Computer Science, PhD in Economics, Associate Professor;
Kapitolina N. Shcherbakova[2],
Head of the Quality Control Sector
Marina V. Glushchenko[3],
Master Student, The Department of Information Systems and Applied Computer Science

MODELING SOFTWARE QUALITY CONTROL PROCESSES

[1, 3] Rostov State University of Economics, Rostov-on-Don, Russia,
[1] kalugyan@yandex.ru, [3] mary_kush@mail.ru
[2] Research Institute “Specvuzavtomatika”, Rostov-on-Don, Russia,
kapitolina-orlova@yandex.ru

Abstract. The issues of quality control of the developed software through testing are considered. The UML language is used for visual modeling of testing processes.

Keywords: software, quality control, testing, modeling.

Необходимость совершенствования процессов обеспечения качества программного обеспечения (ПО) сегодня осознается практически всеми компаниями-разработчиками. Даже те фирмы, которые ранее полагались на удачу в деле обеспечения качества, создают обособленные подразделения контроля качества, налаживают и развивают процессы тестирования.

Среди причин повышения внимания к качеству программного обеспечения отметим обострение конкуренции на рынке прикладного программного обеспечения. Клиенты становятся более «избалованными», они больше не готовы мириться с ошибками и простоем. «Лишь бы считало» – этого больше не достаточно.

Вторая причина – неудовлетворенность сотрудников (разработчиков, специалистов по сопровождению, менеджеров). Даже если потребителя устроит не вполне качественный продукт, производство такого продукта со временем приведет к ухудшению психологического климата в фирме. Не получая морального удовлетворения от качественно выполненной работы разработчики и специалисты по сопровождению либо уйдут, либо станут относиться к работе «спустя рукава».

Наконец, третья причина заключается в том, что не только клиент, но и фирма-разработчик несет убытки от некачественного программного обеспечения. Выпуск «заплаток», взаимодействие с клиентами по поводу их претензий может обойтись дороже, чем профилактическое поддержание системы на приемлемом уровне качества.

На рисунке 1 представлена диаграмма классов для анализа предметной области тестирования программного обеспечения [1].

Традиционное тестирование (в рамках «методологии» “code & fix”) предполагает хаотичное, интуитивное исполнение сотрудником различных функций с целью поиска неисправностей.

В большинстве организаций используются современные методологии тестирования, основанные на понятиях сценария тестирования или тест-кейса, а также соответствующий инструментарий. Такое тестирование предполагает изучение возможностей системы, после чего для тестирования каждой из них строится формальный сценарий. Иногда сценарий делается вырожденным – в виде т.н. чек-листа. Исполнение тестовых сценариев гарантирует гораздо большую вероятность нахождения багов (дефектов). Деятельность по созданию сценариев (тест-дизайн) поручается квалифицированным сотрудникам.

Часть тестовых сценариев может быть автоматизирована с помощью специального программного обеспечения, позволяющих создавать скрипты для исполнения определенных задач.

Класс **тест-кейс** является ключевым для данной модели. Тест-кейс представляет сценарий тестирования. Он призван обеспечивать контроль какой-то определенной функциональной возможности **программы**.

Каждый тест-кейс содержит предварительное условие (что нужно сделать, чтобы начать тест-кейс), описание, код и название, а также уровень важности (например, можно использовать трехуровневую шкалу High, Medium, Low).

Тест-кейсы объединены в группы, которые называются **тест-сьюты**.

Каждый тест-кейс включает **шаги**. Каждый шаг тест-кейса имеет номер, действие и ожидаемый результат.

После того, как выпускается обновленная или новая **версия** программы начинается работа по планированию тестирования. Формируется тестовый **план**. Этой работой занимается старший тестировщик или руководитель группы тестирования. Создание планов – всегда компромисс между желанием максимально проверить программу и ограниченными ресурсами. В обязательном порядке максимально строго проверяется новая или модифицированная функциональность. Но остальные функции, проверенные ранее для предыдущих версий также подвергаются частичной проверки в рамках регрессионного тестирования.

Тестовый план содержит набор тест-кейсов, которые следует оттестировать для данной версии на определенной **конфигурации** оборудования.

Класс **Результат** отражает результат прогона конкретного тест-кейса. Включает **результаты шагов** этого тест-кейса. Для каждого шага фиксируется статус результата – пройден/провален. В последнем случае провален и сам тест-кейс.

В случае провала формируется **Дефект** или иначе «баг». Каждый дефект получает уникальный номер, название, описание (обязательно конфигурацию, условие и т. д.), дату и время создания, уровень критичности (например, High, Medium, Low). Дефект создается тестировщиком, добившемся провала тест-кейса под руководством старшего тестировщика.

Для возможности воспроизведения дефекта программистом он содержит **вложения**: **картинки** снимков экрана, **видео** действий тестировщика, **sql-скрипт** для создания необходимой ситуации в базе данных и т. д.

Далее начинается деятельность по багтрекингу. Начинается «жизнь» дефекта – программист его воспроизводит, начинает исправлять. Иногда ведется обширная переписка с помощью **ответов**. В результате дефект меняет свой статус, вплоть до уровня «закрыт» после проверки тестировщиком. Как правило, исправление дефекта включается в следующую версию ПО. На рисунке 2 показана диаграмма прецедентов для процессов тестирования.

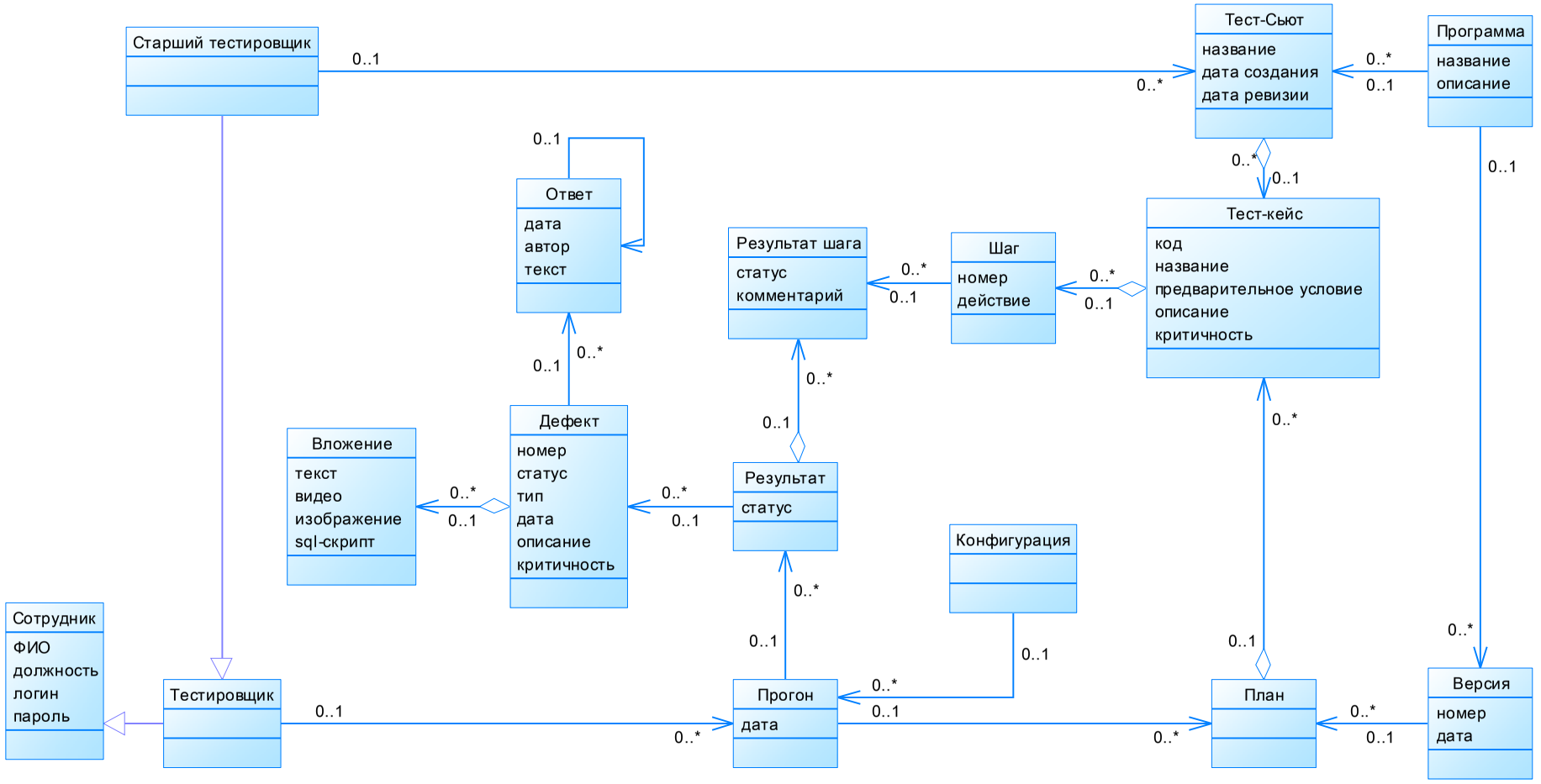

Рис. 1. Диаграмма классов предметной области процессов тестирования

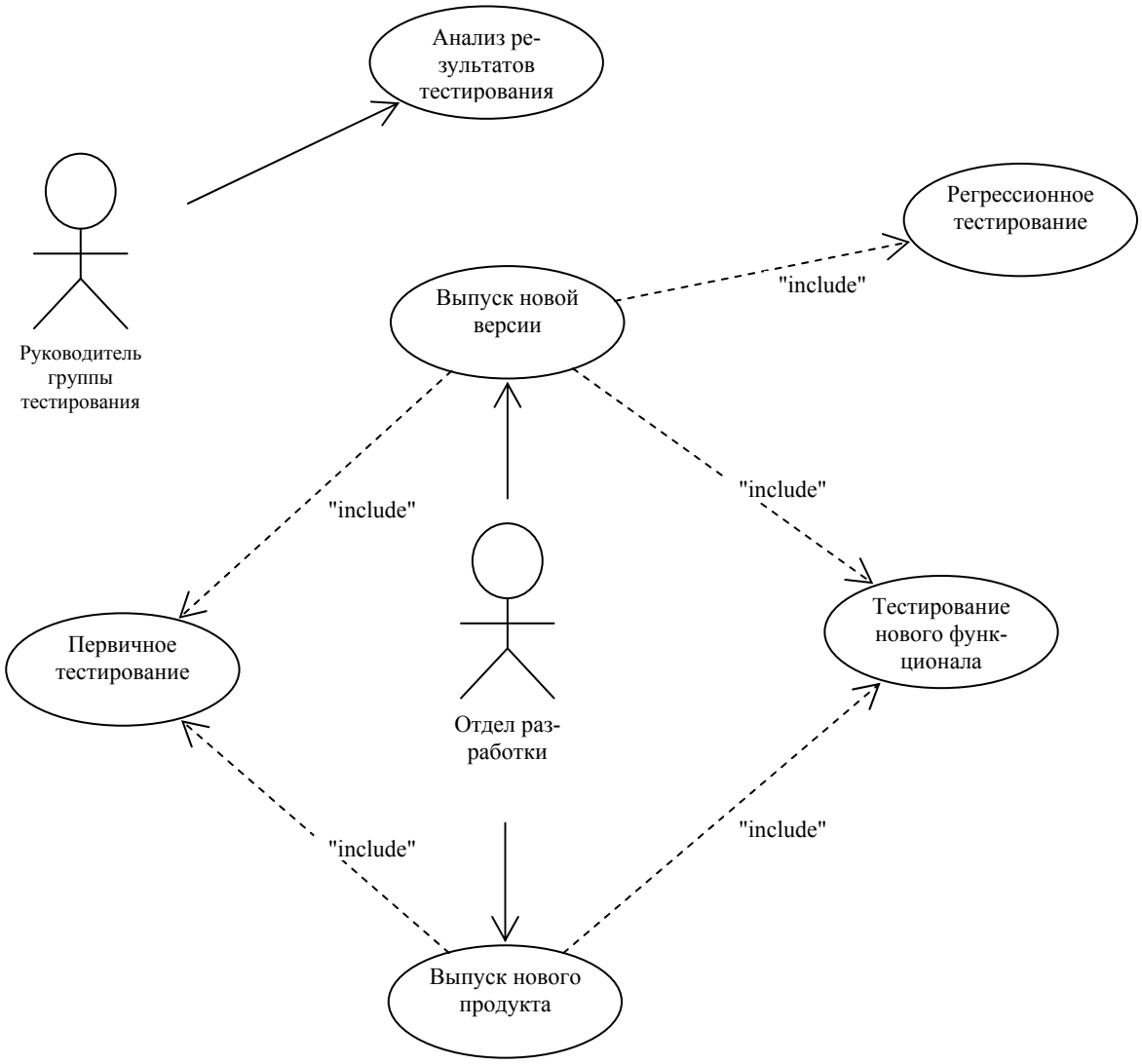

Рис. 2. Диаграмма прецедентов совокупности бизнес-процессов тестирования ПО

Здесь также действует актер **Отдел разработки**, который выпускает новое ПО и новые версии существующего ПО. Каждый такой прецедент заставляет отдел тестирования исполнять соответствующий бизнес-процесс тестирования ПО.

Сначала осуществляется **первичное тестирование** (smoke, дымовое), которое призвано убедиться, что программа запускается и позволяет исполнять основной функционал.

Тестирование нового функционала – основной прецедент, который включает изучение спецификации, подготовку тестовых сценариев (в соответствии с правилами и приемами тест-дизайна), планирование прогонов (исходя из компромисса между желанием тотального тестирования и ограниченными возможностями отдела), прогона тестовых сценариев на разнообразных конфигурациях и вынесение вердикта.

Для новых версий существующего ПО имеется прецедент **Регрессионное тестирование** (связан отношением «include»), то есть повторное тестирование ранее проверенных компонентов.

Актер **Руководитель группы тестирования** (QA Team Lead) здесь выступает и как актер, который осуществляет **анализ результатов** тестирования с определенной периодичностью. Цель – оценка качества ПО и контроль эффективности работы сотрудников.

На рисунке 3 приведен ключевой бизнес-процесс «Тестирование нового функционала». Этот процесс стартует после выдачи нового или модифицированного программного обеспечения. В нем задействованы все сотрудники группы.

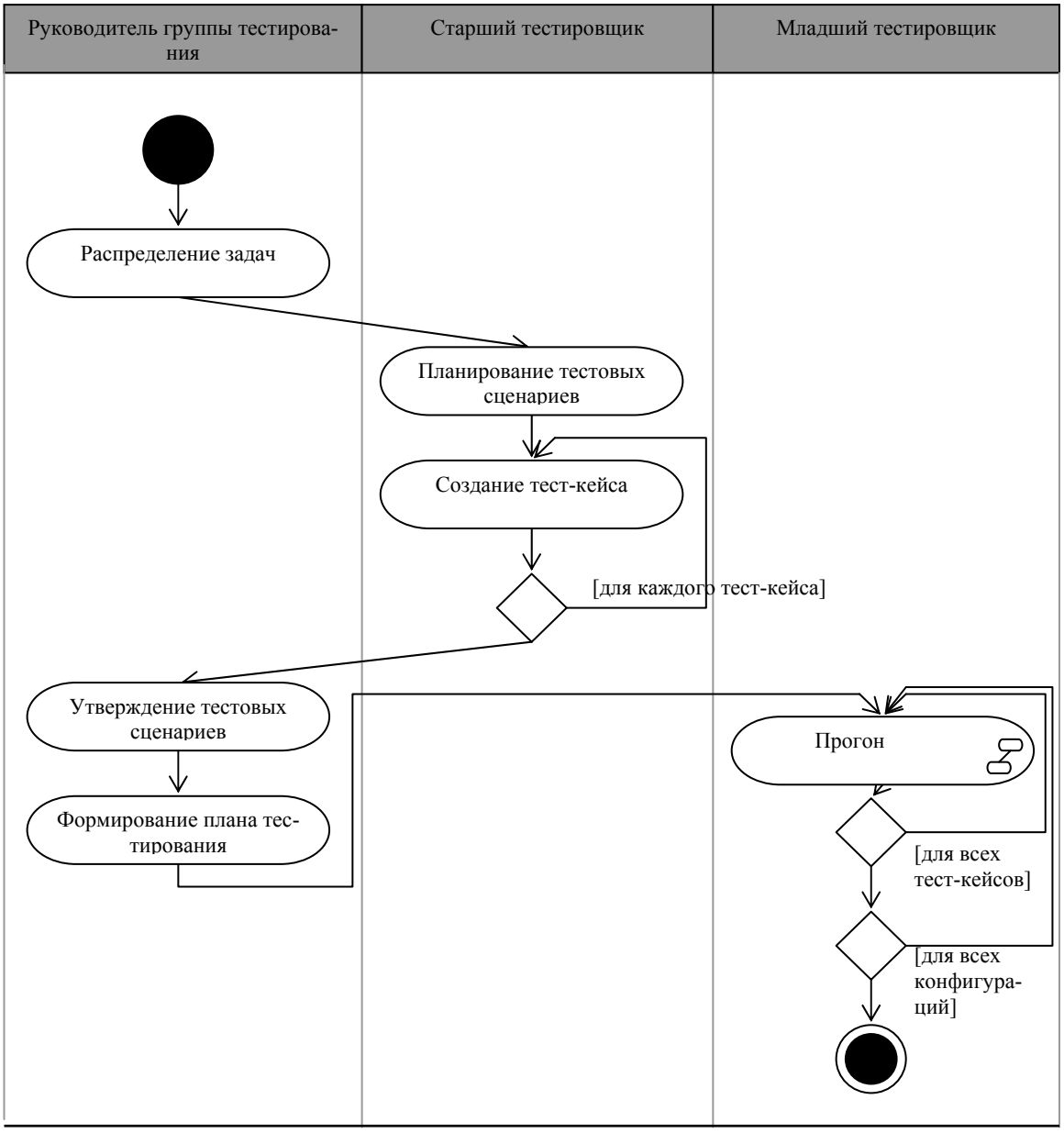

Рис. 3. Бизнес-процесс «Тестирование нового функционала»

Прежде всего, руководитель осуществляет планирование, проводя декомпозицию главной задачи и «разбрасывая» подзадачи между подчиненными. Старший тестировщик приступает к разработке тестового покрытия нового функционала. Планируются тестовые сценарии, которые соответствуют новым функциям и обеспечивают возможности их проверки, причем как по основной, так и по альтернативным веткам их исполнения. Затем прорабатывается каждый сценарий (тест-кейс) в отдельности. Совокупность тест-кейсов передается руководителю на утверждение. Далее создаются тестовые планы, которые и исполняются, как правило, младшим тестировщиком путем прогонов тест-кейсов на всех назначенных конфигурациях оборудования. Если на каком-то шаге возникают отклонения фактического результата исполнения от ожидаемого, то исполнение тест-кейса прекращается. Как шаг, так и тест-кейс считаются проваленными, а тестировщик заводит дефект в системе багтрекинга.

Представленные диаграммы позволили [1]:

1) описать совокупность бизнес-процессов на самом высоком уровне абстракции;

2) описать границы исследуемой системы;

3) выделить актеров и показать их связи с бизнес-процессами;

4) показать, что диаграммы могут служить основой для автоматизированного построения имитационной модели.

Разработанная визуальная UML-модель бизнес-процессов служит основой для построения имитационной модели тестирования программного обеспечения путем ее автоматизированного синтеза [2, 3].

Список литературы

1. Хубаев Г.Н., Калугян К.Х., Родина О.В., Щербаков С.М., Широбокова С.Н. Визуальное и имитационное моделирование деловых процессов для экспресс-оценки ресурсоёмкости товаров и услуг // The scientific heritage. 2016. № 5 (5). С. 92–99.

2. Хубаев Г.Н., Щербаков С.М. Особенности построения и использования системы автоматизированного синтеза имитационных моделей СИМ-UML // Имитационное моделирование: теория и практика» (ИММОД-2015): Материалы Седьмой Всероссийской научно-практической конференции по имитационному моделированию и его применению в науке и промышленности. М., 2015. С. 400–403.

3. Широбокова С.Н., Щербаков С.М. Возможности метода и программного комплекса автоматизированного синтеза имитационных моделей деловых процессов // Труды Междунар. науч.-техн. конф. «Компьютерное моделирование 2008». СПб.: Изд-во Политехн. ун-та, 2008. С. 259–268.