УДК 004.415.53
В. Н. Леванцов, Е. В. Рафалова
г. Гомель, ГГУ имени Ф. Скорины

ПРЕИМУЩЕСТВА И НЕДОСТАТКИ МЕТОДОЛОГИЙ TEST-DRIVEN DEVELOPMENT И BEHAVIOR-DRIVEN DEVELOPMENT

Тестирование играет ключевую роль в обеспечении качества программного обеспечения. Путем выявления и исправления дефектов на ранних стадиях разработки тестирование помогает предотвратить появление ошибок в конечном продукте, улучшая его надежность и стабильность. Контроль качества – обеспечить доверие пользователей к продукту. Тестирование помогает снизить риски, такие как потеря данных, нарушение безопасности, что помогает избежать ущерба репутации компании и юридических последствий.

При подготовке специалистов на степень бакалавра по специальности 6-05- 0612-03 Системы управления информацией в учебном плане прописана необходимость приобретения навыков и практического опыта по тестированию разрабатываемого программного обеспечения.

Тестирование дополнительно ассоциируется с затратами как временными, так и финансовыми, которые также важно учитывать в ходе разработки и тестирования продукта. Один из важных аспектов, который следует учитывать при выборе стратегии тестирования – это стоимость поддержки тестов разного типа в течение жизненного цикла программного обеспечения. Для обеспечения экономической целесообразности в современной разработке применяется концепция пирамиды тестирования, которая предлагает иерархию различных уровней тестирования, балансирующих между стоимостью поддержки, процентом покрытия кода и эффективностью в поиске багов. К уровням тестирования относят модульные, интеграционные и энд-ту-энд (E2E) тесты.

Модульные тесты разрабатываются как тестирование отдельных компонентов: функций, методов, классов. Выполняются эти тесты изолированно от остальных. К преимуществам указанного уровня тестирования относится быстрая обратная связь, а также наименьшая стоимость создания и поддержки актуального состояния тестов. К недостаткам выполнения модульных тестов следует отнести то, что тесты не учитывают взаимодействие классов между собой, и эффективность их ограничивается тем, насколько хорошо определены контракты между классами.

Тестирование взаимодействия между компонентами системы производится в рамках интеграционного тестирования. Данный уровень пирамиды тестирования позволяет проверить работоспособность и ошибки в интеграции между внутренними компонентами системы. На уровне компонентов тестирование внешних по отношению к сервису зависимостей проводится с помощью техник мокирования, что оставляет пространство для ошибки, если мок не повторяет поведения реального сервиса в точности, что является серьезным недостатком интеграционного тестирования.

Системные или сервисные тесты производят тестирование взаимодействия между различными службами или сервисами приложения. На этом уровне происходит проверка работы программных интерфейсов приложения, веб-сервисов, а также интеграции компонентов с внешними системами. Сервисное тестирование требует наличия внешних по отношению к сервису зависимостей, что сопровождается дополнительной стоимостью и комплексной поддержкой.

E2E тесты позволяют провести полное тестирование системы с точки зрения пользователя от входа до выхода, что позволяет проверить работу системы как единого целого, выявить проблемы на уровне пользовательского интерфейса. Но такой подход требует наличия окружения со всеми зависимостями, включая внешние программные интерфейсы, что сопровождается высокими затратами и является целесообразным только для проверки основных сценариев действий пользователя.

Каждый уровень пирамиды тестирования имеет свои сильные и слабые стороны, и эффективное тестирование требует сбалансированного подхода к использованию всех типов тестов. Модульные тесты обеспечивают быструю обратную связь и низкую стоимость, но не улавливают проблемы интеграции. Интеграционные и системные тесты проверяют взаимодействие компонентов и сервисов, но могут быть более сложными в разработке и поддержке. Энд-ту-энд тесты обеспечивают полное покрытие системы, но требуют значительных ресурсов и могут быть медленными в запуске.

Помимо стоимости и трудозатрат, важным пунктом является сама методология написания тестов. Существуют различные методологии, которые отличаются в таких аспектах, как время написания тестов и целеполаганиями в тестировании. Наиболее популярные в современной разработке подходы – это Test-Driven Development (TDD) и Behavior-Driven Development (BDD).

Методология TDD представляет собой подход к разработке программного обеспечения, основанный на создании тестов непосредственно перед написанием самого кода. Этот цикл разработки, в котором тесты пишутся перед реализацией функциональности, позволяет улучшить качество кода, облегчить его тестирование и ускорить процесс разработки, а также поощряет написание лекготестируеомого кода. В методологии TDD разработчик начинает с написания теста, который описывает ожидаемое поведение новой функциональности или исправляет ошибку в существующем коде. Затем разработчик пишет минимальный объем кода, который необходим для прохождения этого теста. Цель данного этапа – сделать тесты пройденными успешно, а не реализация полного функционала. После прохождения теста код проходит процесс рефакторинга, где он улучшается без изменения его поведения. Этот шаг помогает улучшить читаемость, поддерживаемость и расширяемость кода. Затем цикл повторяется, пока функционал не написан целиком. Важными преимуществами TDD являются высокое качество получаемого кода и тестов, где тесты дополняют документацию, что уменьшает вероятность ошибок и улучшает его структуру. В ходе написания функционала разработчики моментально узнают о нарушении существующего функционала при внесении изменений. Недостатками же являются время, необходимое на обучение разработчиков данной методологии, а также относительно высокие затраты на начальные тесты: на первых этапах написание тестов может потребовать больше времени, чем написание самого кода.

Методология BDD базируется на принципах TDD, фокусируя внимание на ожидаемом поведении программы на уровне бизнес-требований. BDD описывает поведение программы через спецификации, которые затем переводятся в тесты на основе поведения. BDD реализуется с помощью специализированного инструментария, например Cucumber или SpecFlow. BDD поощряет коллаборацию между участниками проекта: представителей бизнеса, разработчиков и тестировщиков. Методология помогает перевести абстрактные бизнес-требования в понятное ожидаемое поведение программы. Использование естественного языка позволяет участникам проекта лучше понимать требования и ожидания друг друга, а также способствует сокращению разрыва между пониманием разработчиков и ожиданием бизнеса.

Процесс начинается с определения поведения программы через сценарии использования, известные как UserStories. На основании UserStories пишутся спецификации на естественном языке в виде FeatureFiles, используя ключевые слова, такие как Given, When, Then. Процесс автоматизируется с помощью BDD- фреймворков.

Несмотря на вышеперечисленные преимущества, у BDD есть ряд недостатков: внедрение BDD требует времени и усилий для того, чтобы команда разработки интегрировала методологию в свои процессы. Спецификации в BDD часто требуют поддержки и обновления по мере изменения требований и развития приложения. С увеличением сложности приложения может потребоваться больше времени и усилий для поддержания актуальности и целостности спецификаций. Также важным пунктом является фокус подхода на конечном продукте, и тестирование на уровне классов и компонентов не учитывается методологией BDD.

Перечисленные подходы и методологии позволяют определить оптимальный набор инструментов для достижения главной цели тестирования – подтверждение того, что программный продукт функционирует в соответствии с его задачами и требованиями.

УДК 378.016:378.046.2
В. П. Лемешев
г. Гомель, ГГУ имени Ф. Скорины

ВОПРОСЫ СОВЕРШЕНСТВОВАНИЯ УЧЕБНОЙ РАБОТЫ СТУДЕНТОВ

Основная задача любого образовательного процесса – подготовка профессионально компетентного специалиста, обладающего необходимыми знаниями, умениями и навыками для качественного и производительного труда в рамках своих компетенций. Подразумевается, что такая деятельность неразрывно связана с постоянным самосовершенствованием и развитием умений, приобретённых во время учёбы. Высококвалифицированный специалист таковым является лишь в случае, когда цели и средства его деятельности отвечают современным запросам общества и соответствуют направлениям его социального развития. Поэтому учебная работа студентов носит не только обучающий фактор, но и в достаточно большом объёме состоит из воспитательной и идеологической направленности. Она является основополагающим фактором деятельности каждого вуза. Широкое применение цифровых технологий в обучении приводит к сокращению аудиторной нагрузки. В такой ситуации возрастает роль самостоятельной работы студентов. Особенно это важно для освоения многоуровневых абстрактных математических понятий, которые требуют многократного их повторения для выработки умений использования их в многочисленных практических приложениях. С другой стороны, такие факторы вынуждают преподавателей во всё более оптимальной степени реализовать творческий индивидуальный подход к каждому студенту и осуществлять постоянный объективный контроль его учебной работы. Как правило, не сразу учащиеся осознают необходимость своей успешной учебной деятельности. В этом состоит воспитательный аспект организации их работы для каждого преподавателя вуза. В настоящей работе проанализирован опыт организации учебной деятельности по некоторым математическим дисциплинам.

Анализ организации практической работы по курсам «Геометрия и линейная алгебра» на факультете математики и технологий программирования показывает важность своевременного планирования и постановки требований к учебной работе на предстоящий семестр. Студенты должны помнить, что успехи в учёбе и в их дальнейшем профессиональном карьерном росте носят публичный характер. Поэтому вся их работа в университете является открытой. Это предполагает активное творческое сотрудничество студенческой группы как коллектива с преподавателями и между собой.

Практическая работа по курсу начинается с планирования учебного материала. Это помогает студентам ориентироваться в изучаемых темах и материалах, позволяет им создать список всех учебных материалов и распределить работу над ними в соответствии с учебными программами. При этом устанавливаются конкретные цели и сроки для изучения каждой темы.

В настоящее время по всем дисциплинам указанного выше курса разработаны подробные ЭУМК, включающие весь необходимый теоретический материал и индивидуальные практические задания (проекты) по темам (занятиям) каждому студенту. В ЭУМК можно найти полный список вопросов к экзаменам и зачётам, тренировочные примеры и тесты. Это даёт возможность разработать детальный график защиты проектов для каждого студента и тем самым облегчить планирование своей работы на весь семестр.