ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ГОРОДА МОСКВЫ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА МОСКВЫ
«КОЛЛЕДЖ СВЯЗИ №54»
ИМЕНИ П.М. ВОСТРУХИНА

Отчет по выполнению реферата на тему:

«Инструментарии анализа качества программных продуктов в среде разработке»**.**

Выполнил: студент
группы 2-ИСП 11-19
Маслов Е.С.

Введение

Качество программного продукта определяется по нескольким критериям. Качественный программный продукт должен отвечать функциональным и нефункциональным требованиям, в соответствии с которыми он создавался, иметь ценность для бизнеса, отвечать ожиданиям пользователей .

В жизненном цикле управления приложениями качество должно отслеживаться на всех этапах жизненного цикла ПО. Оно начинает формироваться с определения необходимых требований. При задании требований необходимо указывать желаемую функциональность и способы проверки её достижения.

Качественный программный продукт должен обладать высоким потребительским качеством, независимо от области применения: внутреннее использование разработчиком, бизнес, наука и образование, медицина, коммерческие продажи, социальная сфера, развлечения, веб и др. Для пользователя программный продукт должен удовлетворять определенному уровню его потребностей.

Важным аспектом создания качественного ПО является обеспечение нефункциональных требований, таких как удобство в эксплуатации, надежность, производительность, защищенность, удобство сопровождения. Надежность ПО определяет способность без сбоев выполнять заданные функции в заданных условиях и в течение заданного отрезка времени. Производительность характеризуется временем выполнения заданных транзакций или длительных операций. Защищенность определяет степень безопасности системы от повреждений, утраты, несанкционированного доступа и преступной деятельности. Удобство сопровождения определяет легкость, с которой обслуживается продукт в плане простоты исправления дефектов, внесения корректив для соответствия новым требованиям, управления измененной средой.

Управление жизненным циклом программного продукта помогает разработчикам целенаправленно добиваться создания качественного ПО, избегать потерь времени на переделку, повторное проектирования и перепрограммирование ПО.

Тестирование программного обеспечения

Тестирование программного продукта позволяет на протяжении всего жизненного цикла ПО гарантировать, что программные проекты отвечают заданным параметрам качества. Главная цель тестирования - определить отклонения в реализации функциональных требований, обнаружить ошибки в выполнении программ и исправить их как можно раньше в процессе выполнения проекта.

На протяжении всего жизненного цикла разработки ПО применяются различные типы тестирования для гарантии того, что промежуточные версии отвечают заданным показателям качества. При этом применяются автоматические и ручные тесты.

1. Модульное тестирование - предназначено для проверки правильности функционирования методов классов ПО. Модульные тесты пишутся и исполняются разработчиками в процессе написания кода. Модульное тестирование применяется как для проверки качества кода приложения, так и для проверки объектов баз данных.
2. Исследовательское тестирование - предназначено для тестирования, при котором тестировщик не имеет заранее определенных тестовых сценариев и пытается интуитивно исследовать возможности программного продукта и обнаружить и зафиксировать неизвестные ошибки.
3. Интеграционное тестирование - используется для проверки корректности совместной работы компонентов программного продукта.

4. Функциональное тестирование - предполагает проверку конкретных требований к ПО и проводится после добавление к системе новых функций.
5. Нагрузочное тестирование - предназначено для проверки работоспособности программного продукта при предельной входной нагрузке.
6. Регрессионное тестирование - применяется при внесении изменений в программное обеспечение с целью проверки корректности работы компонентов системы, которые потенциально могут взаимодействовать с измененным компонентом.
7. Комплексное тестирование - предназначено для тестирования функциональных и нефункциональных требований всей системы программного продукта.
8. Приемочное тестирование - представляет собой функциональные испытания, которые должны подтвердить то, что программный продукт соответствует требованиям и ожиданиям пользователей и заказчиков. Приемочные тесты пишутся бизнес-аналитиками, специалистами по контролю качества и тестировщиками.

Для разработчика программного обеспечения в VisualStudio 2012 предоставлена возможность создавать модульные и нагрузочные тесты, а также тесты пользовательского интерфейса. В VisualStudio 2012 имеются следующие шаблоны тестовых проектов:

-проект модульного теста, который позволяет создавать модульные тесты в процессе разработки;

-проект с веб-тестами производительности и нагрузочными тестами;

-проект с закодированными тестами пользовательского интерфейса.

Инструментарием тестировщика в VisualStudio 2012 является MicrosoftTestManager (MTM). MTM предназначен для управления жизненным циклом тестирования программного обеспечения, включая планирование, тестирование и мониторинг. MTM интегрирован с TeamFoundationServer. С помощью Microsoft TestManager тестировщики подготавливают планы тестирования, управляют тестированием. При создании плана тестирования в него добавляются наборы тестов, тестовые случаи и конфигурации, необходимые для тестирования. Конфигурации используются для установления среды, в которой будут исполняться наборы тестов. Microsoft TestManager позволяет выполнять ручные и автоматические тесты, а также исследовательские тесты. Результаты тестирования сохраняются в базе данных, что позволяет подготавливать различные аналитические отчеты. Ошибки, выявленные в процессе тестирования, фиксируются, документируются и передаются разработчикам для их устранения. При внесении изменений в код программной системы возникает необходимость в регрессионном тестировании, причем MTM автоматически формирует план регрессионного тестирования, выявляя какие тесты должны быть повторно выполнены.

Для тестировщиков и разработчиков программного обеспечения VisualStudio 2012 включает диспетчер виртуальной среды LabManagement. Инструментарий тестирования LabManagement позволяет создать инфраструктуру, которая максимально близко эмулирует реальную среду планируемого использования программного продукта. Такие среды могут использоваться для выполнения автоматических построений, автоматизации тестов и выполнения разработанного кода.

Рефакторинг

Качество кода программного продукта во многом определяется тем, насколько трудно или легко вносить изменения в код, а также тем насколько доступен код для понимания. Созданное программное *приложение* может выполнять требуемые функции, но иметь проблемы с внесением изменений или пониманием созданного кода. В этом случае такое *ПО* нельзя назвать качественным, так как на этапе его сопровождения могут возникнуть проблемы с его модификацией при изменении пользовательских требований.

Для улучшения качества кода программных приложений применяют рефакторинг. *По* определению Фаулера М. рефакторинг определяется как "процесс изменения программной системы таким образом, что её внешнее поведение не изменяется, а внутренняя структура улучшается". Следовательно, в процессе проектирования для создания высококачественного программного продукта необходимо не только обеспечить выполнение функциональных требований, но и нефункциональных, в частности, удобства сопровождения, что предполагает возможность и простоту внесения изменений в код, а также возможность легкого понимания созданного кода.

Некачественный *дизайн* кода определяется *по* ряду признаков:

- жесткость - характеристика программы, затрудняющая внесение изменений в код;
- хрупкость - свойство программы повреждаться во многих местах при внесении единственного изменения;
- косность характеризуется тем, что код содержит части, которые могли бы оказаться полезными в других системах, но усилия и риски, сопряженные с попыткой отделить эти части кода от оригинальной системы, слишком велики;
- ненужная сложность характеризуется тем, что программа содержит элементы, которые не используются в текущий момент;
- ненужные повторения, которые состоят в повторяющихся фрагментах кода в программе;
- непрозрачность, которая характеризует трудность кода для понимания.

Для создания качественного дизайна кода целесообразно применять некоторые принципы и паттерны проектирования ПО[41, 42].

Принцип единственной обязанности определяет, что у класса должна быть только одна причина для изменения. Из этого принципа следует, что классу целесообразно поручать только одну обязанность.

Принцип открытости/закрытости определяет, что программные сущности (классы, модули, функции) должны быть открыты для расширения, но закрыты для модификации.

Принцип подстановки Лисков определяет, что должна быть возможность вместо базового класса подставить любой его подтип.

Принцип инверсии зависимостей определяет два положения:

- модули верхнего уровня не должны зависеть от модулей нижнего уровня. И те и другие должны зависеть от абстракций;
- абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.

Принцип разделения интерфейсов определяет, что клиенты должны знать только об абстрактных интерфейсах, обладающих свойством сцепленности.

Паттерны проектирования предлагают универсальные, проверенные практикой решения. В [40] приведен обширный перечень паттернов. Среди общего списка паттернов следует выделить те, которые целесообразно применять при гибкой разработке программного обеспечения. Это паттерны Команда, Стратегия, Фасад, Посредник, Одиночка, Фабрика, Компоновщик, Наблюдатель, Абстрактный сервер/адаптер/шлюз, Заместитель и Шлюз, Посетитель и Состояние.

Использование паттернов при разработке позволяет создавать программное обеспечение, которое легко модифицировать и сопровождать.

Следует отметить, что для проведения рефакторинга необходимо иметь надежные тесты, которые обеспечивают контроль соблюдения функциональных требований при улучшении дизайна кода программного обеспечения.

Краткие итоги

Качество программного продукта определяется по нескольким критериям и качественный программный продукт должен отвечать функциональным и нефункциональным требованиям. В жизненном цикле приложения качество должно отслеживаться на всех его этапах. Тестирование программного продукта позволяет на протяжении всего жизненного цикла ПО гарантировать, что программные проекты отвечают заданным параметрам качества. На протяжении всего жизненного цикла разработки ПО применяются различные типы тестирования. Инструментарием тестировщика в VisualStudio 2012 является MicrosoftTestManager и диспетчер виртуальной среды LabManagement. Для улучшения качества кода программных приложений применяют рефакторинг.