Лекция 1. Основные понятия и определения. Качество программных продуктов

1.1. Тестирование ПО. Этапы развития

Тестирование – это процесс выполнения программ или иная деятельность с программой и программными документами, осуществляемая в целях обнаружения факта наличия ошибок или аттестации программного продукта.

Термины "программа", "программная система", "программное обеспечение", "программные средства" и "программный продукт" часто применяются как слова синонимы. Однако, между ними существует некоторое различие:

Программа − описание алгоритма решения задачи на языке электронной вычислительной машины.

Программное обеспечение (Software) (ПО) − полный набор или часть программ, процедур, правил и связанной с ними документации системы обработки информации. Термин "Software" переводится на русский язык также как программное средство или как программный продукт. Однако, продукт − это предмет, как результат человеческого труда, следствие, результат, порождение чего-либо, а средство − прием, способ, действие для достижения чего-либо, орудие для осуществления какой-нибудь деятельности. Для производителей молоток − продукт, для плотника − средство. Так как в данном случае речь идет о разработке (производстве) программного обеспечения, будем использовать понятие "**программный продукт**" (ПП), если речь идет о процессе создания его, и понятие "**программное средство**" (ПС), если программный продукт применяется как инструментарий для производства других программных продуктов, документации и т.п. (например, среда программирования, отладчик). Переданный пользователю готовый программный продукт также становится средством.

Экземпляр или копию разработанного ПП называют **программным изделием** (ПИ). Изготовление ПИ − это процесс генерации и/или воспроизведения (снятия копии) программ и программных документов ПП с целью их поставки пользователю для применения по назначению. Производство ПИ − это совокупность работ по обеспечению изготовления требуемого количества ПИ установленные сроки. Стадия производства ПИ в жизненном цикле ПС является, по существу, вырожденной (не существенной), так как представляет рутинную работу, которая может быть выполнена автоматически и без ошибок. Этим она принципиально отличается от стадии производства различной техники. В связи с этим в литературе эту стадию, как правило, не включают в жизненный цикл ПС.

Программная система - аппаратное и программное обеспечение. предназначенное для решения каких-либо задач. В соответствии с ГОСТ Р ИСО/МЭК 12207−2010 система (system) − это комплекс, состоящий из процессов, технических и программных средств, устройств и персонала, обладающий возможностью удовлетворять установленным потребностям или целям.

Наряду и в сочетании с термином «тестирование» применяют термин отладка. **Отладка** – это процесс ***локализации*** (определение оператора, в котором допущена ошибка) ***и исправления*** (устранения) ошибок, обнаруженных при тестировании программного обеспечения.

Нередко понятие отладка используется в более *широком* смысле: **отладка** ПП – деятельность, направленная на обнаружение, локализацию и исправление ошибок в программном обеспечении. В этом случае тестирование в целях обнаружения факта наличия ошибок является элементом процесса отладки.

1.2. Этапы развития предметной области «Тестирование ПП»

Понятие тестирования программного обеспечения **менялось** в ходе развития вычислительной техники и программных средств.

В **50–60-х** годах 20-го века процесс тестирования ПО стал самостоятельным. Процесс тестирования ПО базировался на концепции «исчерпывающего тестирования». **Исчерпывающее тестирование** предполагает исполнение всех предусмотренных программой вариантов обработки всех возможных наборов исходных данных, и тем самым гарантирует полную проверку программы. Однако исчерпывающее тестирование в большинстве случаев **невозможно**: даже для простейших программ количество вариантов входных данных оказывается астрономическим. И при этом не гарантируется отсутствие ошибок в требованиях к программному продукту, архитектуре ПП и в прочей документации на него.

В 70-е годы появились две противоположные трактовки «тестирования»:

1) тестирование – это процесс доказательства **работоспособности** (правильности) программы в некоторых заданных условиях (positive testing), это процесс верификации, направленный на демонстрацию корректности ПП, позволяющий удостовериться, что программа соответствует требованиям.

2) тестирование – процесс доказательства **неработоспособности** программы в некоторых заданных условиях (negative testing). Ставится задача поиска ошибок и определения условий, при которых программа ведёт себя некорректно.

Эти противоречивые трактовки отражают две взаимодополняющие цели тестирования. При этом, доказательство неработоспособности считается более продуктивным с точки зрения улучшения качества ПП, так как не позволяет игнорировать обнаруженные проблемы.

В 80-е годы 20-го века тестирование перестало быть только заключительной стадией проекта. Тестирование требований, архитектуры, документации и других элементов проекта позволило во многих случаях воплотить концепцию **предупреждения дефектов** и, как следствие, сократить сроки и бюджет разработки. Также для этого периода характерно появление и развитие **методологий** тестирования, и появление первых инструментов для **автоматизированного** тестирования.

В **90-х** годах тестирование стало элементом более глобального процесса «**обеспечения качества** (quality assurance)». Развиваются методологии тестирования, появляются мощные инструменты управления процессом тестирования и инструменты автоматизации тестирования.

В **2000-х** продолжается поиск новых путей, методологий, техник и подходов к обеспечению качества. Появились подходы, в которых **разработка ПО осуществляется под управлением тестирования** (test-driven development, TDD. Автоматизация тестирования становится неотъемлемой частью проектов.

Сегодня для тестирования характерно применение гибких методологий и автоматизации, глубокая интеграция его с процессом разработки, большой набор технологий и инструментальных средств.

1.3. Место процесса тестирования в жизненном цикле программного продукта (ПП) и в процессе его разработки

Тестирование занимает значительное место в жизненном цикле ПП вообще и процессе разработки программного обеспечения, в частности (до 50% от времени работы над проектом).

Жизненным циклом программного обеспечения (software life cycle) называют период *от момента появления идеи* создания некоторого программного продукта до момента завершения его поддержки фирмой, выполнявшей сопровождение и далее *до прекращения* всех видов его *использования*.

Для представления последовательности процессов жизненного цикла ПП обычно используют 1) каскадную (водопадную), 2) инкрементную и 3) спиральную модели жизненного цикла. В ряде источников (например, [2]) выделяют также 4) компонентно-ориентированную модель. Выбор модели ЖЦ зависит от специфики самой создаваемой системы и условий, в которых она разрабатывается и функционирует.

1) Каскадной (водопадной) называют модель, в рамках которой переход на следующую стадию осуществляется после полного *завершения* проектных *операций* предыдущей стадии и *получения* всех *исходных данных* для следующей стадии. Модель требует определить *опорные точки*, в которых будет оцениваться сделанное и решаться вопрос о том, можно ли двигаться дальше. Модель считается классической.

2) Инкрементная модель объединяет в себе элементы последовательной водопадной модели с итерационной философией макетирования. *Каждый инкремент* (прототип), включающий процессы каскадной модели *завершается созданием работающего программного продукта*, функциональность которого на каждом инкременте увеличивается. Каждый инкремент демонстрируется заказчику. При этом выясняется, какие пожелания заказчика поняты неправильно. Модель применяется для небольших систем.

3) Спиральная модель может рассматриваться как развитие инкрементной модели. Она включает четыре квадранта: 1) *планирование* – определение целей, вариантов, ограничений; 2) *анализ риска*; 3) *конструирование* – разработка продукта очередного уровня и 4) *оценивание* – оценка заказчиком текущих результатов проекта. На каждой итерации строится очередная версия программы и в конце каждого витка имеется работоспособный программный продукт (версия, прототип программы), а каждый очередной процесс разработки предваряет анализ риска. В ходе анализа риска определяют, в каких ситуациях может возникнуть риск, оценивается вероятность возникновения и величина потерь для каждого выявленного для проекта элемента риска.

4) Компонентно-ориентированная модель является развитием спиральной модели. В ней конкретизируется содержание квадранта конструирования, чтобы показать необходимость использования уже существующих программных компонентов в новой разработке.

Основным нормативным документом, регламентирующим жизненный цикл программного обеспечения, является международный стандарт ISO/IEC 12207[3]. Его полный аутентичный текст содержит соответствующий российский стандарт **ГОСТ Р ИСО/МЭК 12207–2010** «Информационная технология. Системная и программная инженерия. Процессы жизненно цикла программных средств» Он устанавливает общую структуру процессов жизненного цикла программных средств, на которую можно ориентироваться в программной индустрии. Стандарт определяет процессы, виды деятельности и задачи, которые используются при приобретении и поставке

программного продукта или услуги, а также при разработке, применении по назначению, сопровождении и прекращении применения программных продуктов. Все ***процессы***, связанные с разработкой программных продуктов в стандарте разбиты на **семь** групп.

В списке встречается **два** процесса тестирования, это процессы квалификационного тестирования системы и квалификационного тестирования программных средств. Цель процесса ***квалификационного тестирования системы*** заключается в подтверждении того, что реализация каждого системного требования тестируется и система ***готова к поставке***. Процесс квалификационного тестирования **программных средств** –элемент процесса реализации. Он выполняется **для каждого компонента программной системы.** Цель процесса – проверить соответствие каждого программного элемента квалификационным требованиям, осуществимость сборки системы и убедиться в том, что комплектованный программный продукт удовлетворяет установленным требованиям. Но и остальные процессы 4-й и 5-й групп включают в свой состав какой-либо вид тестирования. Например, в процессе функционирования программные средства тестируются и настраиваются в предназначенной для них среде. В ходе процесса реализации осуществляют выполнение и документирование (виды и сроки тестирования, исходные данные и результаты, выявленные проблемы и т.п.) испытаний каждого программного модуля и базы данных.

Однако тестирование используется и как **часть процессов других групп**, например, при ***планировании*** разработки определяют место и роль тестирования в общем процессе разработки.

Почти все **процессы поддержки** программных средств связаны с тестированием. Например, процесс обеспечения **гарантии качества** программных средств направлен на обеспечение соответствия установленным стандартам и процедурам, требованиям и планам используемых в жизненном цикле проекта программных продуктов и процессов, процесса измерений и конечного программного продукта. И проверка соответствия – одна из задач тестирования.

В свою очередь, процесс квалификационного тестирования ПП – элемент процессов верификации и валидации программных средств. Необходима верификация требований, проекта, кода, комплексирования, документации. В ходе ***валидации*** осуществляется подтверждение того, что требования выполняются для **конкретного применения** рабочего программного продукта. Помимо тестирования могут использоваться другие средства, такие как анализ, моделирование, имитация и т.п.

Цель процесса ***аудита*** программных средств – гарантии того, что a) реализованные программные продукты отражают проектную документацию; b) условия приемки и требования к тестированию пригодны для приемки программной продукции; c) тестовые данные соответствуют спецификациям; d) программные продукты успешно протестированы и удовлетворяют спецификациям; e) расхождения между фактическими и ожидаемыми результатами устранены и т.п.

В соответствии со стандартом **процесс разработки** (конструирования) охватывает работы по созданию компонентов программного обеспечения и системы в целом в соответствии с заданными требованиями. Модель процесса конструирования обобщает опыт создания программ и показывает последовательность работ по разработке ПП и в значительной степени определяет стратегию разработки. Основными моделями процесса разработки являются 1) каскадная, 2) спиральная и 3) инкрементная.

1) Каскадная модель процесса разработки ПО представлена на рисунке 1.1. Она подходит для проектов, в которых требования легко формулируются с самого начала.

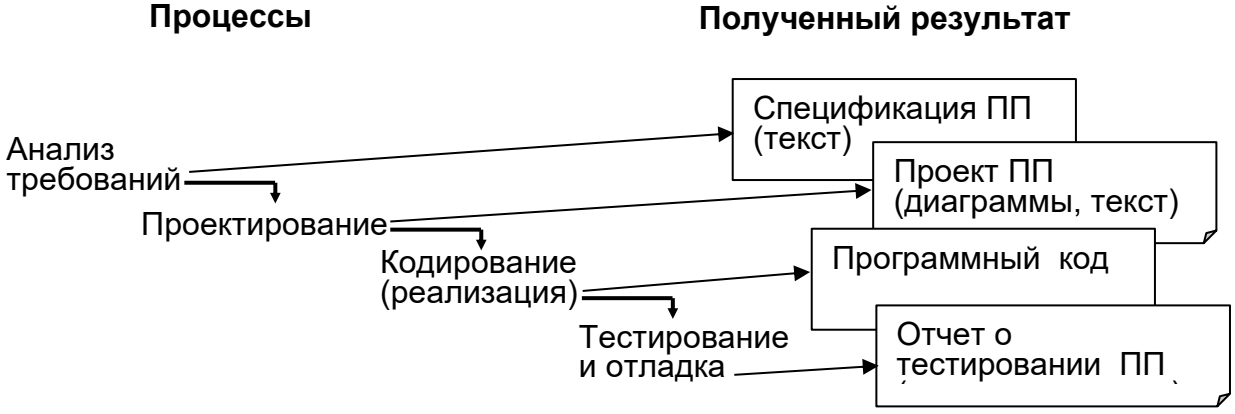

Рисунок 1.1 – Каскадная модель процесса разработки ПО

Основными **достоинствами** каскадной модели являются простота планирования процесса разработки (логичная последовательность этапов позволяет установить сроки завершения всех работ и спланировать затраты на их выполнение), обеспечение условий для высокой эффективности разработки, хорошая документированность ПП. Каждая фаза завершается верификацией и подтверждением, цель которых – устранить возможно большее число проблем (ошибок), связанных с разработкой изделия. Все этапы завершаются выпуском полного комплекта соответствующей проектной документации. Так как очередной этап может выполняться другой командой разработчиков (например, реализация – программистами, тестирование – тестировщиками), комплект должен отвечать требованиям полноты и согласованности достаточной для продолжения проекта.

В чистом виде каскадная модель процесса разработки используется достаточно редко. Более сложные варианты каскадной модели (каскадно-возвратная, каскадно-итерационная, каскадная с подпроцессами и некоторые другие) предполагают возвраты на предыдущие стадии.

На каскадной модели базируется группа водопадных (каскадных) технологических подходов к разработке программных продуктов. Но и большинство других разновидностей процесса разработки берут его за основу.

2) В случае использования **спиральной модели** разработки (рис.1.2) последовательность процессов анализ требований → проектирование → реализация → тестирование выполняется более одного раза.

В соответствии с данной схемой разработка приложения выглядит как серия последовательных итераций. На первых итерациях уточняют спецификации продукта, на последующих – добавляют некоторый ограниченный набор функций, расширяя и наращивая возможности данного продукта. Таким образом. углубляются и последовательно конкретизируются детали проекта. Каждый виток спирали соответствует созданию работоспособного фрагмента или версии ПО.

В силу своей итеративной природы спиральная модель допускает корректировки проекта по ходу работы, что способствует улучшению продукта. Первые версии программного продукта поступают к заказчику через достаточно небольшой промежуток времени. За счет практики использования продукта ускоряется формирование и уточнение спецификаций. Можно собрать на каждом витке спирали метрические характеристики проекта (трудоемкость, затраты на проект, длительность, документированность), оценить риски продолжения работ и уточнить план-график дальнейшей работы.

Рисунок 1.2 – Спиральная модель процесса разработки

В числе основных проблем при использовании спиральной модели, следующие:

- сложность определения моментов завершения очередной стадии. Варианты ее решения: ограничение сроков прохождения каждой стадии, определение критериев достижения контрольных точек;
- необходимость более искусного управления: требует планирование итераций развития ПП;
- необходимость поддержки целостности документации для всех версий.

3) Когда число витков спирали возрастает настолько, что каждая новая итерация предоставляет слишком малое количество новых возможностей по сравнению с предыдущей, то такую модель процесса разработки называют **инкрементальной** (инкрементной) разработкой**.** В этом случае обновление исходного проекта (документации, набора тестов, программного кода и т.д.) осуществляется в течение заданного (сравнительно короткого) интервала времени: недели, дня, нескольких дней. Например, обновления программного кода и документации предоставляются **ежедневно** к конкретному времени для интеграции и ночного тестирования. Модель требует четко установленной архитектуры проекта и синхронизации системы документации.

Во всех перечисленных моделях тестирование показано, как самостоятельный процесс. В реальных условиях работы связанные с тестированием есть фактически в каждом из перечисленных в моделях процессе. Наиболее ярко это показывает **V-модель** процесса разработки (рис.1.3): тестируются пользовательские требования, архитектура, компоненты, код, сборка.

Рисунок 1.3 – V-образная модель процесса разработки ПП

1.4. Качество программных продуктов

В настоящее время особое внимание уделяется вопросам обеспечения качества программных средств на протяжении всего их жизненного цикла.

Качеством называют совокупность признаков и характеристик продукции, процесса или услуги, которые обеспечивают способность их удовлетворить установленные или предполагаемые потребности.

Тестирование играет важную **роль** в повышении качества, надежности и эффективности ПП, оказывает значительное влияние на его востребованность на рынке программного обеспечения и стоимость разработки.

Разработчик создает то, что продается, обеспечивая материальное вознаграждение за свой труд. Однако людям свойственно ошибаться. *Пользователю* же безразлично, какую квалификацию имеет разработчик, какие паттерны были применены в продукте, который он купил, как много и упорно разработчик оптимизировал свой продукт. Ему важно, чтобы выявившиеся в процессе эксплуатации ПП ошибки не мешали бы ему использовать этот ПП для решения своих задач.

Конечно, плохой проект отличное тестирование вряд ли спасет. Но изначально плохой проект может и не появиться, если тестировщики включатся в процесс разработки не ПОСЛЕ, а ДО создания ПП, если на этапе обсуждения задачи (Product-owner) они начнут задавать свои «глупые вопросы». Ответы на вопросы тестировщика дают команде понимание того, что некоторые требования (сценарии, решения, алгоритмы и т.п.) не настолько просты и очевидны, как кажется на первый взгляд, что на их проработку и реализацию следует выделить больше ресурсов (человеко-часов) и возможно пересмотреть их приоритет.

V-модель процесса разработки показывает необходимость применения тестирования на ВСЕХ его этапах. В связи с этим есть мнение, что тестер должен быть **более квалифицированным**, чем программист: тестер должен понимать, как устроен весь продукт, т.к. он может столкнуться с проблемой в любом месте и должен уметь с ней разобраться. Чем раньше найден дефект – тем меньше цена его исправления.

Низкое качество ПП влечет за собой:

а) финансовые потери:

- возможное уменьшение продаж продукта;
- возможное возмещение убытков покупателям;
- увеличение расходов на службу поддержки;
- судебные издержки;
- потеря имиджа компании;

б) потери времени:

- поиск причин проблемы;
- исправление кода;
- повторное тестирование продукта.

Экономические последствия низкого качества программного обеспечения могут быть весьма огромны. Примеры **«дорогих»** ошибок:

– Ошибка в управляющем коде рентгеновского аппарата «Therac-25» привела к смерти 5 пациентов (1980)

– Веерное отключение электричества в Северной Америке из-за ошибки в управляющем ПО (2003)

– крушение из-за ошибки ПО самолета Боинг в Эфиопии, погибло 157 человек (2019).

Не столь ярко выражены ***косвенные потери*** от неудовлетворительного качества программных продуктов. Финансовых вложений требуют:

– содержание в постоянной готовности *служб технической* поддержки и сопровождения, тестирования, консультирования. Численность таких служб может превышать количество программистов, непосредственно занятых созданием продукта;

– повседневные усилия по ликвидации причин выявленных дефектов и претензий пользователей программных средств;

– устранение проблем, возникших при модернизации ПП, поддержание необходимого уровня их качества.

На требования к качеству ПП и процессы его оценки оказывают **влияние** следующие **факторы**:

- область его применения и назначение (например, медицина: поддержка жизнедеятельности оперируемого; или медицина: ведение базы данных пациентов);
- функциональность, размер и сложность программной системы;
- величина допустимого ущерба из-за недостаточного качества ПП;
- степень связи решаемых задач с реальным масштабом времени или допустимой длительностью ожидания результатов решения задачи;
- предполагаемый срок эксплуатации;
- планируемый набор версий и ожидаемый тираж;
- необходимый уровень документирования.

Выбор показателей качества зависит от того, с чьих позиций рассматривается качество ПП:

1) **Представление пользователя** о качестве связано с назначением ПП, его возможностью удовлетворять заданные или подразумеваемые потребности пользователя.

2) **Представление разработчика** должно включать характеристики качества программного обеспечения, выбранные пользователем. Как правило, именно их реализация в соответствии с техническим заданием является условием подписания акта передачи ПП заказчику. Однако, качество конечной продукции обеспечивается качеством компонентов, соблюдением технологий разработки и т.п. Соответственно одни и те же характеристики на разных этапах разработки оцениваются посредством различных метрик. *Например*, эффективность в терминах времени выполнения программы (на конечном этапе) зависит от числа осуществляемых алгоритмом элементарных операций, затрат времени на обращения к памяти и разрешение конфликтов доступа к памяти и т.п.

3) **Представление руководителя** проекта будет отражать коммерческие требования для характеристик ПП**.** Его задача оптимизировать качество в пределах ограничений стоимости, трудовых ресурсов и сроков, соблюдение плановых показателей.

Также характеристики качества можно разделить на **внутренние и внешние.**

1. **Внешние** метрики – это характеристики ПП, важные для пользователя. К ним относятся**:** корректность (правильность, обеспечивает правильную работу на правильных данных), устойчивость (способность обрабатывать ситуации незапланированные проектом без потери критических данных), защищенность (определяет степень безопасности системы от повреждений, утраты, несанкционированного доступа и преступной деятельности), переносимость (на другие аппаратные и программные средства), легкость использования, совместимость и некоторые другие. В конечном итоге именно внешние характеристики действительно имеют значение.

2**. Внутренние** характеристики пользователя не заботят: не важно, насколько замечательную модульную конструкцию или т.п. вы использовали при создании программы. Однако, они являются *ключом и причиной внешних* характеристик качества. К ним относятся: верифицируемость (простота проверки, легкость разработки тестов, легкость локализации ошибок), читабельность кода, модульность, структурированность, документированность (кода) и другие.

Вопросам стандартизации, оценки качества и сертификации программного обеспечения посвящены несколько национальных, государственных и международных **стандартов**. В их числе ГОСТ **28195**-89 «Оценка качества программных средств. Общие положения», ГОСТ Р ИСО/МЭК **9126-93** «Информационная технология. Оценка программной продукции. Характеристика качества и руководства по их применению», ГОСТ **28806**-90 «Качество программных средств. Термины и определения».

Во всех стандартах общие (комплексные) **характеристики качества** (их еще называют критериями качества) описываются с помощью подхарактеристик (атрибутов, примитивов качества). Примитивами качества называют стандартизованный набор элементарных свойств ПП, однозначно интерпретируемых разработчиками. Один примитив может использоваться для описания нескольких критериев.

Стандарты являются основой, *нормативной базой* для определения требований к качеству конкретного программного продукта. Однако стандарты не дают рекомендаций по выбору критериев качества и разработчик должен самостоятельно разрабатывать систему показателей качества и выявлять влияющие факторы.