КАФЕДРА ИВС

Тема: «Качество ПО и методы его контроля »

Дисциплина: «Верификация, стандартизация и сертификация ПО»

Лектор: старший преподаватель кафедры ИВС Савченко Н.К.

План лекции:

1. Качество программного обеспечения.
2. Методы контроля качества.
3. Ошибки в ПО.

1 Качество программного обеспечения

Понятие качественного ПО соответствует представлению о том, что программа достаточно успешно справляется со всеми возложенными на нее задачами и не приносит проблем ни конечным пользователям, ни их начальству, ни службе поддержки, ни специалистам по продажам.

Качество программного обеспечения определяется в стандарте ISO 9126 как вся совокупность его характеристик, относящихся к возможности удовлетворять высказанные или подразумеваемые потребности всех заинтересованных лиц.

При рассмотрении качества ПО различаются понятия *внутреннего качества*, связанного с характеристиками ПО самого по себе, без учета его поведения, *внешнего качества*, характеризующего ПО с точки зрения его поведения, и *качество ПО при использовании* в различных контекстах – то качество, которое ощущается пользователями при конкретных сценариях работы ПО. Для всех этих взглядов на качество введены метрики, позволяющие оценить его.

Кроме того, при создании качественного ПО существенно *качество технологических процессов* его разработки. Взаимоотношения между этими аспектами качества по схеме, принятой в ISO 9126, показано на рисунке 4.1.

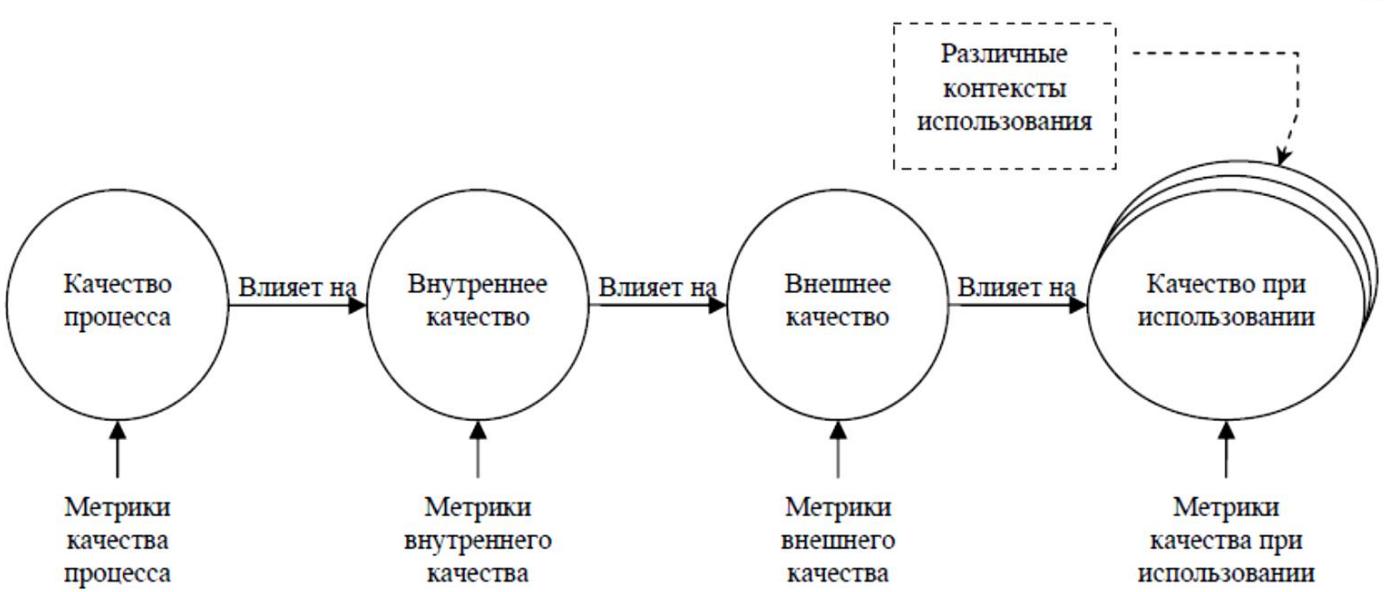

Рисунок 4.1 – Основные аспекты качества ПО по ISO 9126

Общие принципы обеспечения качества процессов производства во всех отраслях экономики регулируются набором стандартов ISO 9000. Наиболее важные для разработки ПО стандарты в его составе следующие.

- **ISO 9000:2000 Quality management systems – Fundamentals and vocabulary** Системы управления качеством — Основы и словарь.
- **ISO 9001:2000 Quality management systems – Requirements. Models for quality assurance in design, development, production, installation, and servicing.**

Системы управления качеством – Требования. Модели для обеспечения качества при проектировании, разработке, коммерциализации, установке и обслуживании. Определяет общие правила обеспечения качества результатов во всех процессах жизненного цикла.

Помимо поддержки и развития системы процессов, нацеленных на удовлетворение нужд заказчиков и пользователей, ISO 9001 требует

- Определить, документировать и развивать собственную систему качества на основе измеримых показателей.
- Использовать эту систему качества в качестве средства управления процессами, нацеливая их на большее удовлетворение нужд заказчиков, планируя и постоянно отслеживая качество результатов всех видов деятельности, в том числе и самого управления.

- Обеспечить использование качественных ресурсов, качественного (компетентного, профессионального) персонала, качественной инфраструктуры и качественного окружения.
- Постоянно контролировать соблюдение требований к качеству на практике, во всех процессах проектирования, производства, предоставления услуг и при приобретениях.
- Предусмотреть процесс устранения дефектов, определить и контролировать качество результатов этого процесса.

- **ISO 9004:2000 Quality management systems – Guidelines for performance improvements**. Системы управления качеством. Руководство по улучшению деятельности.
- **ISO/IEC 9000-3:2004 Software engineering – Guidelines for the application of ISO 9001:2000 to computer software .** Руководящие положения по применению стандарта ISO 9001 при разработке, поставке и обслуживании программного обеспечения.

Этот стандарт конкретизирует положения ISO 9001 для разработки программных систем, с упором на обеспечение качества при процессе проектирования. Он также определяет некоторый набор техник и процедур, которые рекомендуется применять для контроля и обеспечения качества разрабатываемых программ.

Стандарт ISO 9126 предлагает использовать для описания внутреннего и внешнего качества ПО многоуровневую модель. На верхнем уровне выделено 6 основных характеристик качества ПО. Каждая характеристика описывается при помощи нескольких входящих в нее *атрибутов*. Для каждого атрибута определяется набор метрик, позволяющих оценить этот атрибут.

Ниже приведены определения этих характеристик и атрибутов по стандарту ISO 9126:2001.

- ✓ **Функциональность (functionality).** Способность ПО в определенных условиях решать задачи, нужные пользователям. Определяет, что именно делает ПО, какие задачи оно решает.
  - Функциональная пригодность (suitability). Способность решать нужный набор задач.
  - Точность (accuracy). Способность выдавать нужные результаты.
  - Способность к взаимодействию (interoperability). Способность взаимодействовать с нужным набором других систем.

- Соответствие стандартам и правилам (compliance). Соответствие ПО имеющимся индустриальным стандартам, нормативным и законодательным актам, другим регулирующим нормам.
- Защищенность (security). Способность предотвращать неавторизированный, т.е. без указания лица, пытающегося его осуществить, и не разрешенный доступ к данным и программам.

- ✓ **Надежность (reliability).** Способность ПО поддерживать определенную работоспособность в заданных условиях**.**

• Зрелость, завершенность (maturity). Величина, обратная к частоте отказов ПО.

• Устойчивость к отказам (fault tolerance) Способность поддерживать заданный уровень работоспособности при отказах и нарушениях правил взаимодействия с окружением.

• Способность к восстановлению (recoverability). Способность восстанавливать определенный уровень работоспособности и целостность данных после отказа, необходимые для этого время и ресурсы.

• Соответствие стандартам надежности (reliability compliance).

✓ **Удобство использования (usability) или практичность**. Способность ПО быть удобным в обучении и использовании, а также привлекательным для пользователей.

- Понятность (understandability). Показатель, обратный к усилиям, затрачиваемым пользователями, чтобы воспринять набор понятий, на которых основано ПО, и их применимость для решения своих задач.
- Удобство обучения (learnability). Показатель, обратный к усилиям, затрачиваемым пользователями чтобы научиться работе с ПО.
- Удобство работы (operability). Показатель, обратный к усилиям, предпринимаемым пользователями, чтобы решать свои задачи с помощью ПО.
- Привлекательность (attractiveness). Способность ПО быть привлекательным для пользователей. Этот атрибут добавлен в 2001.
- Соответствие стандартам удобства использования (usability compliance).

✓ **Производительность** *(efficiency) или* **эффективность.** Способность ПО при заданных условиях обеспечивать необходимую работоспособность по отношению к выделяемым для этого ресурсам. Можно определить ее и как отношение получаемых с помощью ПО результатов к затрачиваемым на это ресурсам.

• Временная эффективность (time behaviour). Способность ПО выдавать ожидаемые результаты, а также обеспечивать передачу необходимого объема данных за отведенное время.

• Эффективность использования ресурсов (resource utilisation). Способность решать нужные задачи с использованием определенных объемов ресурсов определенных видов. Имеются в виду такие ресурсы, как оперативная и долговременная память, сетевые соединения, устройства ввода и вывода, и пр.

• Соответствие стандартам производительности (efficiency compliance).

- ✓ **Удобство сопровождения (maintainability)**. Удобство проведения всех видов деятельности, связанных с сопровождение программ.

- Анализируемость (analyzability) или удобство проведения анализа. Удобство проведения анализа ошибок, дефектов и недостатков, а также удобство анализа на предмет необходимых изменений и их возможных эффектов.
- Удобство внесения изменений (changeability). Показатель, обратный к трудозатратам на проведение необходимых изменений.
- Стабильность (stability). Показатель, обратный к риску возникновения неожиданных эффектов при внесении необходимых изменений.
- Удобство проверки (testability). Показатель, обратный к трудозатратам на проведение тестирования и других видов проверки того, что внесенные изменения привели к нужным эффектам.
- Соответствие стандартам удобства сопровождения (maintainability compliance).

- **Переносимость (portability).** Способность ПО сохранять работоспособность при переносе из одного окружения в другое, включая организационные, аппаратные и программные аспекты окружения. Иногда эта характеристика называется в русскоязычной литературе мобильностью.
  - Адаптируемость (adaptability). Способность ПО приспосабливаться к различным окружениям без проведения для этого действий, помимо заранее предусмотренных.
  - Удобство установки (installability). Способность ПО быть установленным или развернутым в определенном окружении.
  - Способность к сосуществованию (coexistence). Способность ПО сосуществовать с другими программами в общем окружении, деля с ним ресурсы.
  - Удобство замены (replaceability) другого ПО данным. Способность ПО использоваться вместо другого ПО для решения тех же самых задач в заданном окружении.
  - Соответствие стандартам переносимости (portability compliance).

Для описания качества ПО при использовании стандарт ISO 9126-4 предлагает следующий набор характеристик.

- Эффективность (effectiveness). Это способность ПО предоставлять пользователям возможность решать их задачи с необходимой точностью при использовании в заданном контексте.
- Продуктивность (productivity). Способность ПО предоставлять пользователям определенные результаты в рамках ожидаемых затрат ресурсов.
- Безопасность (safety). Способность ПО обеспечивать необходимо низкий уровень риска нанесения ущерба жизни и здоровью людей, бизнесу, собственности или окружающей среде.
- Удовлетворенность (satisfaction). Способность ПО приносить удовлетворение пользователям при использовании в заданном контексте.

Помимо перечисленных характеристик и атрибутов качества стандарт ISO 9126:2001 определяет наборы метрик для оценки каждого атрибута. Приведем следующие примеры таких метрик.

- Полнота реализации функций — процент реализованных функций по отношению к перечисленным в требованиях. Используется для измерения функциональной пригодности.
- Корректность реализации функций — правильность их реализации по отношению к требованиям. Используется для измерения функциональной пригодности.
- Отношение числа обнаруженных дефектов к прогнозируемому. Используется для определения зрелости.
- Отношение числа проведенных тестов к общему их числу. Используется для определения зрелости.
- Отношение числа доступных проектных документов к указанному в их списке. Используется для измерения удобства проведения анализа.
- Наглядность и полнота документации. Используется для оценки понятности.

2 Методы контроля качества

Эффективность верификации и валидации, как и эффективность разработки ПО в целом зависит от точности и корректности формулировки требований к программному продукту.

Основой любой системы обеспечения качества являются методы его обеспечения и контроля. *Методы обеспечения качества* представляют собой техники, гарантирующие достижение определенных показателей качества при их применении.

Методы контроля качества предназначены для того, чтобы убедиться, что определенные характеристики качества ПО достигнуты. Сами по себе они не могут помочь их достижению, они лишь помогают определить, удалось ли получить в результате то, что хотелось, или нет, а также найти ошибки, дефекты и отклонения от требований. Методы контроля качества ПО можно классифицировать следующим образом.

- Методы и техники, связанные выяснением свойств ПО во время его работы. Это, прежде всего, все виды *тестирования*, а также *профилирование* и измерение количественных показателей качества, которые можно определить по результатам работы ПО — эффективности по времени и другим ресурсам, надежности, доступности и пр.
- Методы и техники, связанные с определением показателей качества на основе симуляции работы ПО с помощью моделей разного рода. К этому виду относятся *проверка на моделях (model checking)*, а также *прототипирование (макетирование)*, использованное для оценки качества принимаемых решений.

- Методы и техники, предназначенные для выявления нарушений формализованных правил построения исходного кода ПО, проектных моделей и документации. К методам такого рода относится *инспектирование кода*, заключающееся в целенаправленном поиске определенных дефектов и нарушений требований в коде на основе набора шаблонов, автоматизированные методы поиска ошибок в коде, не основанные на его интерпретации, методы проверки документации на согласованность и соответствие стандартам.
- Методы и техники, связанные с обычным или формализованным анализом проектной документации и исходного кода для выявления их свойств.

К этой группе относятся многочисленные методы *анализа архитектуры ПО*, о которых пойдет речь в следующей лекции, методы формального доказательства свойств ПО и формального анализа эффективности применяемых алгоритмов.

- Обычно при помощи проверки свойств на моделях анализируют два вида свойств использованных при построении ПО алгоритмов. *Свойства безопасности (safety properties)* утверждают, что нечто нежелательное никогда не случится в ходе работы ПО. *Свойства живости (liveness properties)* утверждают, наоборот, что нечто желательное при любом развитии событий произойдет в ходе его работы.

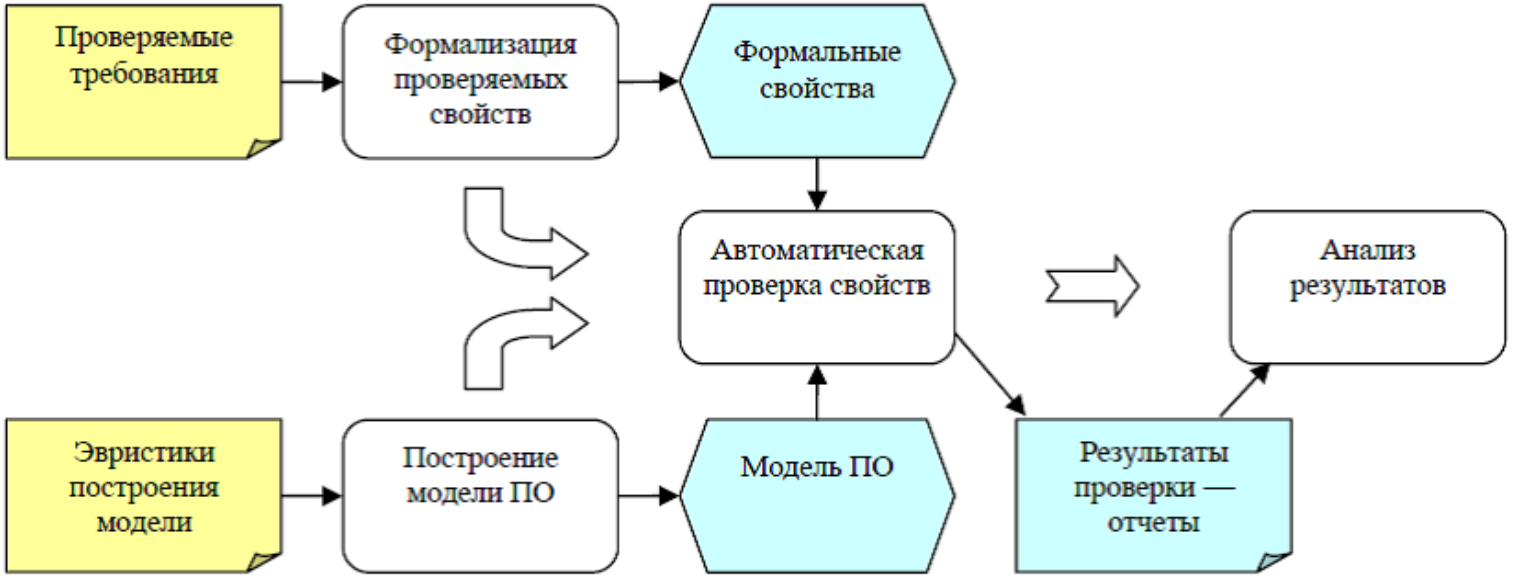

Рисунок 4.2 - Схема процесса проверки свойств ПО на моделях

Примером свойства первого типа служит отсутствие *взаимных блокировок (deadlocks).*

Взаимная блокировка возникает, если каждый из группы параллельно работающих в проверяемом ПО процессов или потоков ожидает прибытия данных или снятия блокировки от одного из других, а тот не может этого сделать, потому что не может продолжить выполнение, ожидая того же от первого или от третьего процесса, и т.д.

Примером свойства живости служит гарантированная доставка сообщения, обеспечиваемая некоторыми протоколами – как бы ни развивались события, если сетевое соединение между машинами будет работать, посланное с одной стороны (процессом на первом ПК) сообщение будет доставлено другой стороне (процессу на втором ПК).

Основная проблема этого подхода — огромное количество состояний в моделях, достаточно хорошо отражающих поведение реальных программ.

Для борьбы с комбинаторным взрывом состояний используются многочисленные методы оптимизации представления автомата и оптимизации выделения и поиска существенных для выполнения проверяемого свойства состояний.

3 Ошибки в ПО

Ошибка (error) – это действие человека, которое порождает неправильный результат.

Однако программы разрабатываются и создаются людьми, которые также могут допускать (и допускают) ошибки. Это значит, что недостатки есть и в самом программном обеспечении. Они называются дефектами или багами (оба обозначения равносильны). Здесь важно помнить, что программное обеспечение – нечто большее, чем просто код.

Дефект, Баг (Defect, Bug) – недостаток компонента или системы, который может привести к отказу определенной функциональности. Дефект, обнаруженный во время исполнения программы, может вызвать отказ отдельного компонента или всей системы.

При исполнении кода программы дефекты, заложенные еще во время его написания, могут проявиться: программа может не делать того, что должна или наоборот – делать то, чего не должна, – происходит сбой.

Сбой (**failure**) – несоответствие фактического результата (actualresult) работы компонента или системы ожидаемому результату (expectedresult).

Сбой в работе программы может являться индикатором наличия в ней дефекта.

Таким образом, сбой существует при одновременном выполнении трех условий:

- известен ожидаемый результат;
- известен фактический результат;
- фактический результат отличается от ожидаемого результата.

Причиной сбоев могут быть не только дефекты, но также и условия окружающей среды: например, радиация, электромагнитные поля или загрязнение также могут влиять на работу как программного, так и аппаратного обеспечения.

Всего существует несколько источников дефектов и, соответственно, сбоев:

- ошибки в спецификации, дизайне или реализации программной системы;
- ошибки использования системы;
- неблагоприятные условия окружающей среды;
- умышленное причинение вреда;
- потенциальные последствия предыдущих ошибок, условий или умышленных действий.

Дефекты могут возникать на разных уровнях, и от того, будут ли они исправлены и когда, будет напрямую зависеть качество системы.

Качество (Quality) – степень, в которой совокупность присущих характеристик соответствует требованиям.

Качество программного обеспечения (Software Quality) – это совокупность характеристик программного обеспечения, отражающих его способность удовлетворять установленные и предполагаемые потребности.

Требование (Requirement) – потребность или ожидание, которое установлено. Обычно предполагается или является обязательным. Условно, можно выделить пять причин появления дефектов в программном коде.

1. Недостаток или отсутствие общения в команде. Зачастую, бизнес требования просто не доходят до команды разработки. У заказчика есть понимание того, каким он хочет видеть готовый продукт, но если должным образом не объяснить его идею разработчикам и тестировщикам, результат может оказаться не таким, как предполагалось. Требования должны быть доступны и понятны всем участникам процесса разработки ПО.

2. Сложность программного обеспечения. Современное ПО состоит из множества компонентов, которые объединяются в сложные программные системы. Многопоточные приложения, клиент-серверная и распределенная архитектура, многоуровневые базы данных – программы становятся все сложнее в написании и поддержке, и тем труднее становится работа программистов. А чем труднее работа, тем больше ошибок может допустить исполняющий ее человек.

3. Изменения требований. Даже незначительные изменения требований на поздних этапах разработки требуют большого объема работ по внесению изменений в систему. Меняется дизайн и архитектура приложения, что, в свою очередь, требует внесения изменений в исходный код и принципы взаимодействия программных модулей. Такие текущие изменения зачастую становятся источником трудноуловимых дефектов.

4. Плохо документированный код. Сложно поддерживать и изменять плохо написанный и слабо документированный программный код. Во многих компаниях существуют специальные правила по написанию и документированию кода программистами. Хотя на практике часто бывает так, что разработчики вынуждены писать программы в первую очередь быстро и это сказывается на качестве продукта.

5. Средства разработки ПО. Средства визуализации, библиотеки, компиляторы, генераторы скриптов и другие вспомогательные инструменты разработки – это тоже зачастую плохо работающие и слабо документированные программы, которые могут стать источником дефектов в готовом продукте.

Пример классификации ошибок

Вид ошибки	Причина	Пример
Требований	Неполное описание требований	Не предусмотрена функция восстановления пароля
Проектирования	Неверная архитектура	Неправильное разделение модулей
Программирования	Ошибка в коде	Использован неверный оператор сравнения
Интеграции	Несовместимость компонентов	Неверный формат обмена данными между сервисами
Производительности	Неэффективный алгоритм	Длительная обработка больших объемов данных
Безопасности	Недостаточная защита	SQL-инъекция через форму ввода
Интерфейса	Ошибка отображения	Некорректное расположение кнопок
Эксплуатации	Ошибка настройки	Неверная конфигурация сервера

Контрольные вопросы для СРС:

1. **Что понимается под качеством программного обеспечения?** Охарактеризуйте основные характеристики качества ПО в соответствии со стандартом **ISO/IEC 25010** (функциональная пригодность, надежность, производительность, безопасность, удобство использования, сопровождаемость и др.).
2. **Какие виды ошибок могут возникать при разработке программного обеспечения?** Классифицируйте ошибки (ошибки требований, проектирования, программирования, интеграции, эксплуатации и др.), укажите причины их возникновения и методы обнаружения.
3. **Разработайте план обеспечения качества для программного модуля.** Определите характеристики качества, методы контроля качества, возможные виды ошибок и способы их предотвращения и устранения на различных этапах жизненного цикла программного обеспечения.

Список рекомендуемой литературы

1. В.В. Кулямин. Методы верификации программного обеспечения. - М.: Институт системного программирования РАН, 2019. – 211 с.

2. В.В. Липаев. Программная инженерия. – М.: Гос. ун-т - Высшая школа экономики. – ТЕИС, 2016. – 608 с.

3. С.В. Синицын. Верификация программного обеспечения: учебное пособие. – Саратов: Профобразование, 2019. — 368 с.

Лекция окончена.

Спасибо за внимание!