8. Тестирование

8.1. Тестирование, валидация, верификация

«Сомневайся во всём»
Рене Декарт

Чисто психологически тестирование выглядит как формальный и нетворческий вид деятельности, сопровождающий процесс разработки ПС, с которым поневоле приходится мириться. Тестировщики считаются низшей кастой программистов, неспособных к написанию собственного кода. Однако с усложнением программного проекта роль тестирования возрастает, а также увеличивается число видов деятельности, именуемых тестированием, и число подлежащих тестированию артефактов.

Особой роли тестирования в программной инженерии способствуют сложность объекта проектирования и отсутствие рамочных законов верификации проекта в целом (см.1.1,1.2).

Тестирование является весьма многоаспектной темой. В широком смысле к ней можно отнести все виды деятельности, связанной с **испытанием** артефактов процесса проектирования.

Тестирование – целенаправленная деятельность, направленная на выявление ошибок в различных артефактах проекта (цели, требования, модель предметной области или анализа, архитектура, код, сборка, релиз, документация) и в проекте в целом.

Тестирование как соответствие требованиям

Одним из положений о структуре тестировании является тезис о соответствии иерархии требований к системе и проверяющих эти требования тестов (рис.8-1).

Рис. 8-1. V-образная модель тестирования

Здесь во главу угла ставятся требования, на что можно резонно возразить: «А что гарантирует нам качество (валидность, непротиворечивость, обоснованность) самих требований?»

Тестирование на различных этапах жизненного цикла программы

В V-образной модели речь идет об исполнении тестов на готовом продукте (релизе или прототипе). Но тестированию может подлежать и любой другой артефакт процесса разработки, например, видение проекта, модель предметной области или классов анализа. Естественно, что тесты в этом случае будут выглядеть как умозрительные испытания (инспекции) артефакта на основе разнообразных внешних воздействий. Например, модель классов анализа можно тестировать с помощью сценариев исполнения функционала: при выполнении шагов сценария проверяется, могут ли быть извлечены из модели связанные данные, необходимые для исполнения сценария.

Поэтому вторым аспектом тестирования является его привязка к жизненному циклу программы и процессу проектирования. Здесь все просто: берем любой артефакт и добавляем к нему слово «тестирование».

- тестирование модели предметной области (фаза исследования);
- тестирование требований (фазы исследования и развития);
- тестирование прецедентов и сценариев (фаза развития);
- тестирование модели классов анализа (фаза развития);
- тестирование архитектурного прототипа (фаза развития);
- тестирование модели графического интерфейса пользователя (GUI) (фаза развития);
- тестирование документации (фаза развертывания).

И наконец, готовый программный продукт, как самый важный артефакт, подлежит многообразному тестированию в зависимости от проверяемых характеристик или устанавливаемых фактов на этапе **системного тестирования**:

- приемочное тестирование при передаче заказчику;
- установочное тестирование для проверки правильности инсталляции;
- альфа- и бета-тестирование при пробной эксплуатации продукта (внутри организации или на группе пользователей);
- тестирование на соответствие требованиям спецификаций;
- тестирование надежности и устойчивости к сбоям;
- регрессионное тестирование – повторное тестирование после внесения изменений;
- тестирование производительности;
- нагрузочное тестирование (стресс-тестирование) – тестирование при повышенной рабочей нагрузке;
- сравнительное тестирование различных версий;
- восстановительное тестирование для проверки работоспособности после восстановления;
- конфигурационное тестирование;
- тестирование Useability – пользовательских характеристик системы.

Тестирование, валидация, верификация

Тестирование является одним из инструментов контроля **качества ПО,** наряду с **валидацией и верификацией.** Взаимоотношения между ними такие.

В самом простом варианте акцент делается на уровнях проверяемых артефактов проекта (рис.8-2).

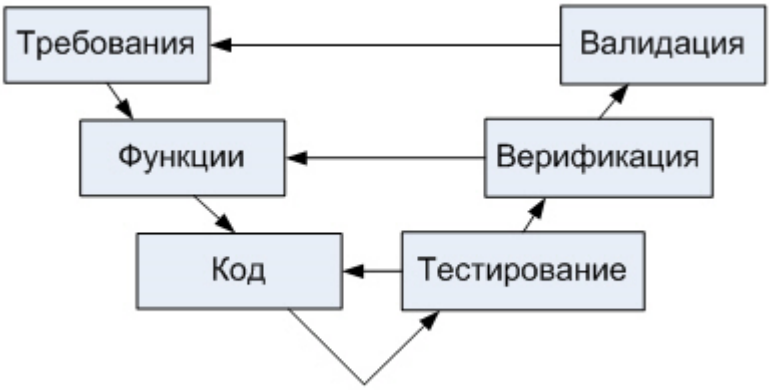

Рис. 8-2. Валидация, верификация, тестирование как уровни испытаний

Приведенная схема не говорит о существе деятельности, которая лежит в их основе. Рассмотрим ее подробнее.

В основе контроля качества ПО лежат валидация и верификация. **Верификация –** это проверка системы на соответствие требованиям, а **валидация** – проверка, насколько сами требования и система соответствуют целям, поставленным при ее создании. Валидация относится в основном к артефактам бизнес-анализа и фазы исследования (бизнес-требования, видение проекта), и касается полезности проекта в целом.

Структуру верификации удобно рассматривать в системе координат «анализ-испытание» и «код - остальные артефакты» (рис. 8-3). Тогда можно определить принципиальную разницу между тестированием и остальными методами верификации следующим образом:

- тестирование - испытание экземпляра артефакта на заданных входных данных (сценарий, тестовый набор), анализ поведения, проверка на ошибки;
- верификация - формальный или содержательный анализ артефакта как сущности.

В каком-то смысле это соответствует принятым в UML понятиям **классификатор и экземпляр.**

Валидация – образно-содержательный анализ соответствия бизнес-требований к системе целям ее создания.

Верификация – проверка на соответствие разработки функциональным требованиям, стандартам качества, анализ на предмет непротиворечивости.

Формальная верификация – способ формального доказательства правильности программного кода путем анализа утверждений о состоянии данных и их преобразовании операциями и операторами программы.

Тестирование – воспроизведение поведения артефакта при заданном входе (испытание);

Тестирование кода (или тестирование в узком смысле) – исполнение готового кода (прототипа или релиза) на тестовых данных или рабочей нагрузке.

Рис. 8-3. Взаимоотношение валидации, верификации и тестирования

Основные методы верификации [8-8] перечислены на рис.8-4. При этом все они применимы к программному коду, остальные артефакты жизненного цикла могут быть проверены только с помощью экспертизы:

- *экспертиза* – «ручной» анализ и оценка артефактов специалистами:
  - *техническая экспертиза* - систематический анализ артефактов проекта квалифицированными специалистами для оценки их внутренней согласованности, точности, полноты, соответствия стандартам и принятым в организации процессам, а также соответствия друг другу и общим задачам проекта;
  - *сквозной контроль* - метод экспертизы, в рамках которого один из членов команды проверки представляет ее участникам последовательно все характеристики проверяемого артефакта, а они анализируют его, задавая вопросы, внося замечания, отмечая возможные ошибки, нарушения стандартов и другие дефекты;
  - *инспекция* - последовательное изучение характеристик артефакта по определенному плану, с целью обнаружения в нем ошибок и дефектов;
  - *аудит* – сторонний анализ артефактов с целью исключения субъективных оценок или привлечения опытных специалистов;
- *статический анализ* – структурный анализ кода на предмет поиска потенциальных угроз и ошибок. Элементы статического анализа кода могут включаться в среды разработки и компиляторы, а также в семантику языка. В Java к таким средствам относятся: проверка переменных на инициализируемость, проверка на достижимость участков кода, контроль перехвата исключений и т.п.;

- *формальные методы* - способы формального доказательства правильности программного кода путем анализа утверждений о состоянии данных (предикаты) и их преобразовании операциями и операторами программы.
- *динамические методы* – основаны на исполнении программного кода и включают в себя:
  - *мониторинг* – непосредственное наблюдение за исполнением;
  - *профилирование* – сбор данных об используемых ресурсах, состоянии программы и окружения во время ее исполнения;
  - *тестирование* – исполнение программы на тестовых данных, по формальным признакам также относится к динамическим методам верификации.

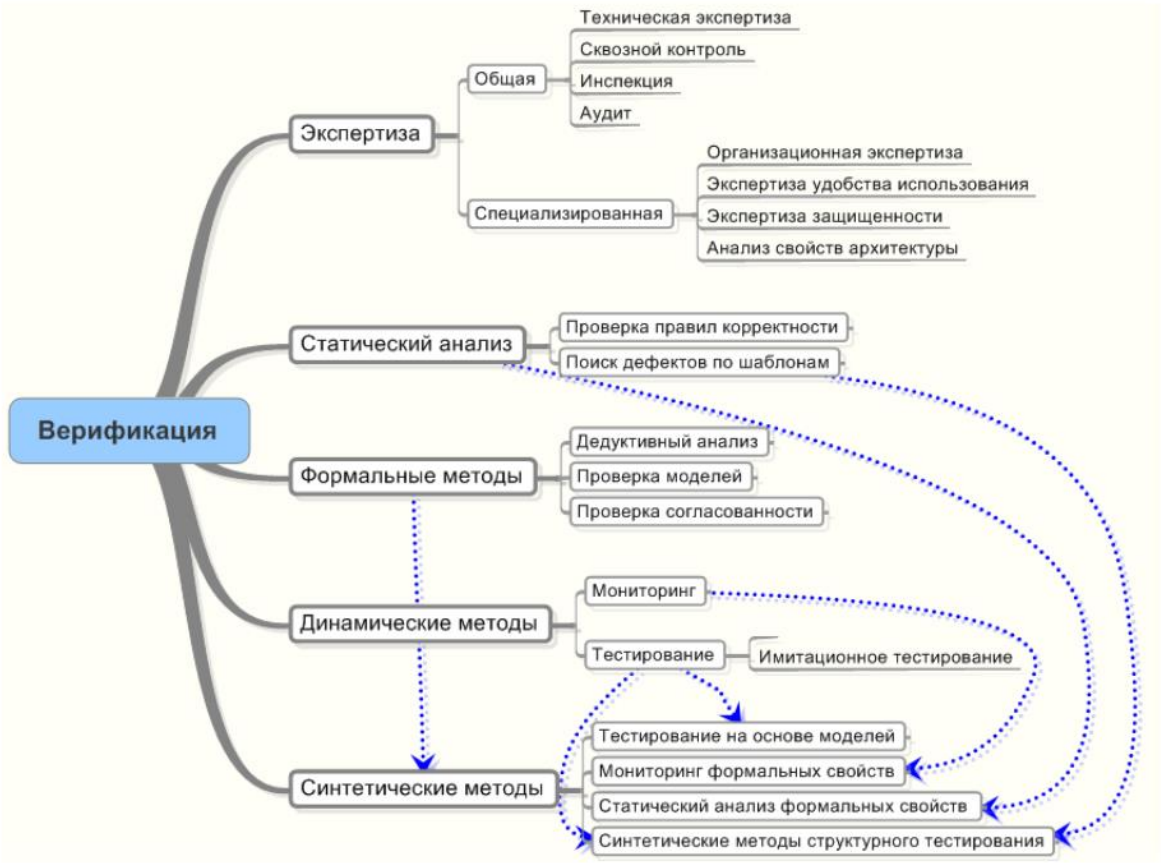

Рис. 8-4.Методы верификации программного обеспечения.

На житейском уровне к верификации кода можно отнести проверку утверждений, формулируемых для *соглашений по данным* и *инвариантам*:

- утверждения о состоянии структур данных (**соглашения по данным).** При проектировании структур данных оговариваются допустимые конфигурации элементов и их значений, которые обязаны соблюдаться всеми компонентами программного кода, работающими с ними (например, всеми методами, класса). Соглашения по данным должны минимизировать возможные сочетания представлений данных. Типичный пример: пустая строка в Си может быть представлена NULL-указателем а также указателем на строку с единственным байтом - символом конца строки;
- утверждения об условиях, сохраняемых на последовательных шагах алгоритма (**инварианты).** При конструировании и тестировании циклов и рекурсивных

алгоритмов весьма продуктивным является формулировка условий, сохраняемых на каждом шаге цикла или рекурсивного алгоритма. Обычно это согласованный набор значений переменных или формальных параметров.

Эмоции и ничего личного. В основе тестирования лежит сомнение или занудство: действительно ли артефакт является идеальным с точки зрения ошибок. Настрой разработчика и тестировщика прямо противоположен: первый уверен в своей непогрешимости, второй стремится ее развенчать. Это противоречие разрешается путем вовлечения разработчика в процесс тестирования, чтобы сделать тестирования составной частью процесса разработки кода (см.8.6).

В заключение процитируем [8-7] перечень видов тестирования (рис.8-5, 8-6).

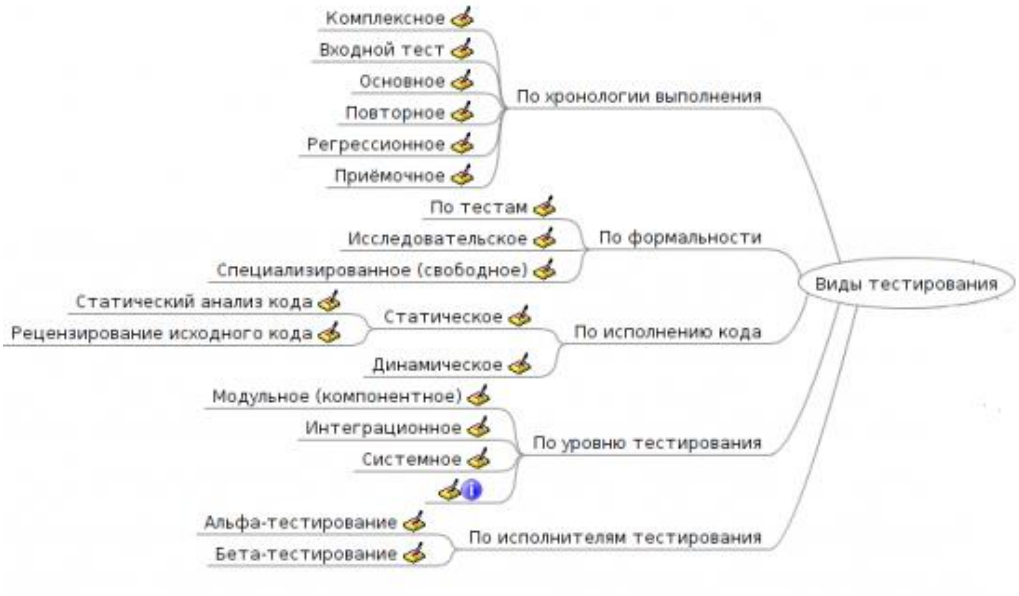

Рис. 81-05. Виды тестирования

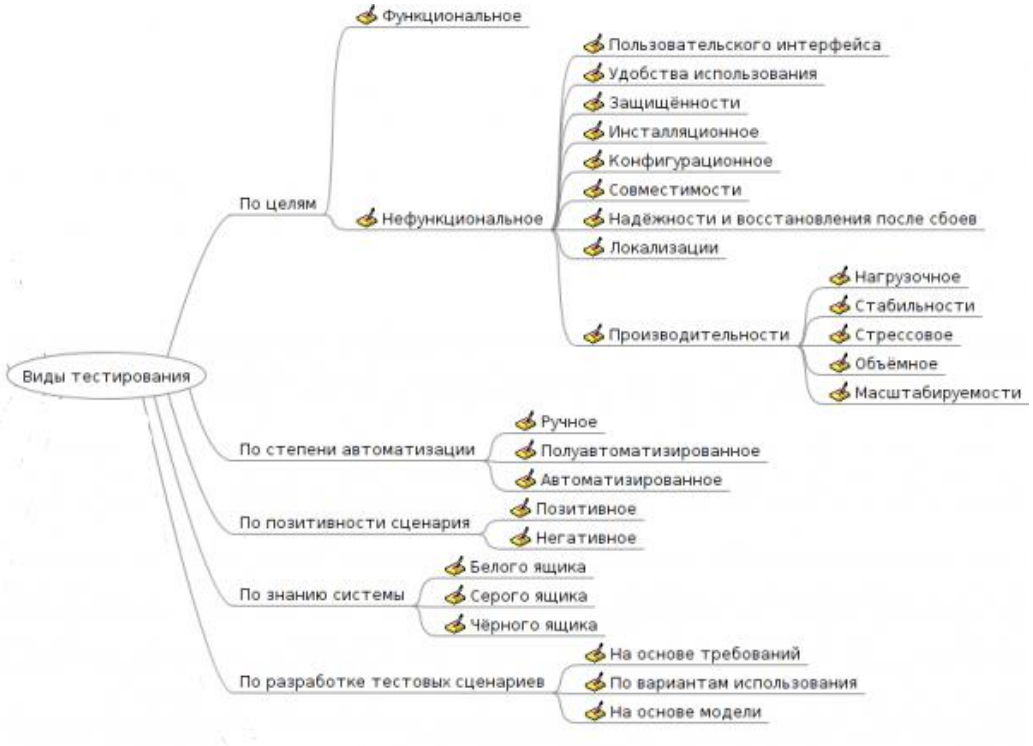

Рис. 8-6. Виды тестирования (продолжение)