Объектно-ориентированное тестирование ИС

Объектно-ориентированный подход накладывает свой отпечаток и на методику тестирования. Вместо тестирования функций, процедур и прочих программных модулей, **производится тестирование классов, связей между ними и иерархий классов.**

Количество классов, их строение и взаимодействие между ними **определяются на стадии объектно-ориентированного проектирования**. На этой стадии создаются диаграммы классов и диаграммы связей. Зачастую по этим диаграммам генерируется исходный программный код в виде абстрактных классов и/или интерфейсов. После этого уже можно провести полноценное тестирование полученной модели. Исправление ошибок на такой ранней стадии позволит значительно сэкономить время отладки и всей разработки в целом. Более того, тестирование поможет выяснить, соответствует ли модель требованиям, предъявленным к программному продукту, и своевременно исправить проект.

Тестирование объектно-ориентированных программ чаще всего выполняется снизу-вверх. Вначале проверяется программный код методов класса и тестируются отдельные методы. Затем начинается тестирование класса.

Для тестирования класса пишется тестовая программа, в которой создаются объекты класса с разными значениями его полей. Тестовая программа проверяет выполнение контрактов методами классов. Для этого она обращается ко всем методам класса и отслеживает результаты их выполнения. При проверке работы методов, обращающихся к другим объектам, **создаются объекты-заглушки**, содержащие только поля и методы, нужные для тестирования основного объекта. Поля объекта-заглушки получают определенные, хорошо узнаваемые значения. Методы объекта-заглушки очень просты, они только сигнализируют каким-нибудь образом обо всех обращениях к ним.

Тестирование шаблонов классов требует особого внимания. Здесь легко допустить ошибку и трудно обнаружить ее. Разработчик должен в точности знать, какой класс создаст компилятор по шаблону, написанному им, а для этого нужен большой опыт работы с этим компилятором. Тестировщик тоже должен знать особенности компилятора, чтобы создать тесты для каждого класса, создаваемого по шаблону.

После того как тестирование отдельных классов уже не выявляет ошибок в них, создаются объекты разных классов, и проверяется их

взаимодействие. **Выстраивается иерархия классов и тестируется правильность наследования в этой иерархии.** Здесь удобнее стиль тестирования сверху-вниз, от базовых классов к порожденным классам, от вершины иерархии классов к самым конкретным классам. Для тестирования абстрактных классов специально создаются его реализации-заглушки.

Особую трудность вызывает тестирование классов, использующих полиморфизм. **Каждый объект такого класса обладает своим поведением**, отличным от поведения других объектов того же типа. **Тип объекта маскируется**, класс полиморфного объекта трудно определить, поэтому нелегко проверить контракт объекта, тем более что в процессе разработки такие объекты могут легко заменяться другими объектами того же типа. **При составлении тестов приходится учитывать будущее поведение полиморфного объекта**, а для этого надо хорошо знать разрабатываемый проект.