# Подходы монолитного, инкрементального, нисходящего и восходящего тестирования

**Монолитное, инкрементальное, нисходящее и восходящее тестирование** — это подходы к интеграционному тестированию программного обеспечения, которые различаются стратегией объединения модулей и подходом к проверке системы.

**Монолитное тестирование**

**Характеризуется одновременным объединением всех модулей в тестируемый комплекс**. После разработки всех модулей они интегрируются, и система проверяется как единое целое.

**Особенности:**

- Для замены неразработанных к моменту тестирования модулей (кроме самого верхнего) требуется разработка драйверов (test driver) и/или заглушек (stub), которые замещают отсутствующие модули нижних уровней.
- **Преимущества:** простота настройки и выполнения, быстрая обратная связь по системным проблемам, упрощённая отладка.
- **Недостатки:** большие трудозатраты на разработку драйверов и заглушек, сложность идентификации ошибок, проявляющихся в пространстве собранного кода, трудности с распараллеливанием работ на начальной фазе тестирования.

**Инкрементальное тестирование**

**Предполагает пошаговое (помодульное) наращивание комплекса программ с пошаговым тестированием собираемого комплекса**. Это позволяет локализовать ошибки на ранних этапах.

В инкрементальном методе выделяют две стратегии добавления модулей:

- **«Сверху вниз»** — соответствует нисходящему тестированию.
- **«Снизу вверх»** — соответствует восходящему тестированию.

**Нисходящее тестирование (Top-Down Integration)**

**Начинается с тестирования модулей верхнего уровня иерархии системы** (пользовательского интерфейса или основной бизнес-логики) и постепенно движется вниз. Поскольку на начальных этапах нижележащие модули ещё не готовы, их функциональность имитируется с помощью заглушек.

**Процесс:**

1. Определяются модули верхнего уровня.
2. Для всех вызываемых ими, но ещё не реализованных модулей создаются заглушки.
3. Тестируется логика верхнего уровня с использованием этих заглушек.
4. По мере готовности реальных модулей они постепенно заменяют соответствующие заглушки, и тесты повторяются.

**Преимущества:** ранняя проверка ключевых бизнес-сценариев и архитектурной целостности системы.

**Недостатки:** проблема разработки достаточно «интеллектуальных» заглушек, сложность организации среды для реализации исполнения модулей в нужной последовательности, возможность не всегда эффективной реализации модулей из-за подстройки ещё не протестированных модулей нижних уровней к уже оттестированным модулям верхних уровней.

**Восходящее тестирование (Bottom-Up Integration)**

**Является противоположностью нисходящего подхода.** Начинается с тестирования самых низкоуровневых модулей (например, библиотек утилит,

слоёв доступа к данным) и движется вверх по иерархии. Поскольку на начальных этапах вышестоящие модули ещё не существуют, для их имитации используются драйверы.

**Процесс:**

1. Низкоуровневые модули проходят модульное тестирование.
2. Для их вызова разрабатываются драйверы.
3. Модули объединяются в кластеры или подсистемы и тестируются с помощью драйверов.
4. По мере готовности вышестоящих модулей они заменяют драйверы, и процесс повторяется на следующем уровне иерархии.

**Преимущества:** гарантия того, что фундаментальные компоненты системы надёжны и работают корректно, прежде чем на их основе будет строиться более сложная логика. Локализация ошибок проще, чем при нисходящем подходе.

**Недостатки:** необходимость в разработке драйверов и заглушек для модульного тестирования перед интеграционным тестированием, а также при интеграционном тестировании части модулей системы.

Иногда применяется **гибридный (сэндвич) подход**, который сочетает нисходящий и восходящий методы тестирования. При этом система делится на слои, тестирование начинается параллельно с верхнего и нижнего слоёв, а финальная интеграция происходит на среднем слое.