**Интеграционное тестирование (в общем случае)**- это вид тестирования, при котором проверяется взаимодействие модулей между собой, а также интеграция подсистем в одну общую систему.

Для интеграционного тестирования используются компоненты, уже проверенные с помощью модульного тестирования.

Модули соединяются между собой с помощью так называемых интерфейсов. **Интерфейс** - это граница между двумя функциональными модулями, например:

* Между двумя модулями одной подсистемы может быть налажено взаимодействие через интерфейс, который определяет способы взаимодействия модулей между собой.
* Между клиентом и сервером (согласно клиент-серверной архитектуре) реализуется интерфейс, с помощью которого передаются данные от клиента серверу и обратно.
* Между двумя различными приложениями многосоставной системы налаживается "общение" с помощью интерфейсов.

*Основная цель интеграционного тестирования - проверить интерфейсы между модулями.*

Важно понимать, что в рамках интеграционного тестирования не проверяются end-to-end бизнес сценарии.

Так как в процессе тестирования у нас нет потребности рассматривать внутреннюю структуру каждого компонента в отдельности, можно утверждать, что интеграционное тестирование выполняется методом «черного ящика».

**Уровни интеграционного тестирования**

Различают два основных уровня интеграционного тестирования:

1. Компонентный;
2. Системный.

На компонентном уровне интеграционного тестирования проверяется взаимодействие между компонентами системы после проведения компонентного тестирования. Другими словами, проверяется, насколько корректно взаимодействуют протестированные в отдельности модули между собой.

На системном уровне проверяется взаимодействие между разными системами после проведения системного тестирования.

**Подходы к интеграционному тестированию**

**Снизу вверх (Bottom Up Integration)**

Все низкоуровневые модули, процедуры или функции собираются воедино и затем тестируются. После чего собирается следующий уровень модулей для проведения интеграционного тестирования. Это продолжается до тех пор, пока не будут интегрированы все модули и конечная система не образует единый модуль.

В случае, представленном на изображении выше, модули B1C1, B1C2, B2C1 и B2C2 являются самыми "низкими" модулями и протестированы отдельно друг от друга с помощью модульного тестирования. Модули B1 и B2 еще не разработаны. В связи с тем, что разработка модулей B1 и B2 находится в процессе, то для тестирования необходима программа, которая обращалась бы к функциям модулей B1C1 и B2C2. Такие программы называются драйверами и представляют собой функции, которые обращаются к функциям более низких уровней. Драйверы необходимы для того, чтобы с помощью интерфейсов вызывать в рамках тестирования более низкие модули.

Данный подход считается полезным, если все (или практически все) модули разрабатываемого уровня готовы. Также данный подход помогает определить уровень готовности приложения по результатам тестирования.

Подход "Снизу-Вверх" позволяет обнаружить дефекты на ранних этапах и позволяет просто локализовать сами дефекты и причины их возникновения.

Недостатком такого подхода является то, что приходится тестировать модули еще до того, как будет реализована "главная" программа, что, соответственно, требует технических навыков.

**Сверху вниз (Top Down Integration)**

Вначале тестируются все высокоуровневые модули, и постепенно, один за другим, добавляются низкоуровневые. Все модули более низкого уровня симулируются заглушками с аналогичной функциональностью, затем, по мере готовности, они заменяются реальными активными компонентами. Таким образом, мы проводим тестирование сверху вниз.

**Большой взрыв ("Big Bang" Integration)**

Все (или практически все) разработанные модули собираются вместе в виде законченной системы или ее основной части и затем проводится интеграционное тестирование. Другими словами, тестирование начинается от середины схемы модулей (для картинки выше) и двигается в обе стороны одновременно.

Такой подход очень хорош для сохранения времени. Однако если тест кейсы и их результаты записаны не верно, то сам процесс интеграции сильно осложнится, что станет преградой для команды тестирования при достижении основной цели интеграционного тестирования. Так же данный подход требует больше ресурсов, в связи с его сложностью.

В целом, для проведения хорошего интеграционного тестирования необходимо:

1. Знать архитектуру приложения;
2. понимать назначение модулей;
3. понимать, как данные передаются из одного модуля в другой;
4. определить условия, при которых происходит взаимодействие между модулями;
5. разрабатывать отдельные тесты на каждое из условий.