ЛАБ-2-Защита

ВАЖНО!
Написано @kkettch по лекциям Клименкова и иногда с помощью ChatGPT, использовалась для защиты ЛР. Информация может оказаться старой или недостоверной.

1. Цели и задачи интеграционного тестирования. Расположение фазы интеграционного тестирования в последовательности тестов; предшествующие и последующие виды тестирования ПО.

- Проверяет интерфейсы и взаимодействие модулей (компонент) или систем
  - Вызовы API, сообщения между ОО компонентами
  - Баз Данных, пользовательский графический интерфейс
  - Интерфейсы взаимодействия (сетевые, аппаратные, локальные, ….)
  - Инфраструктурные
- Может проводиться когда два компонента разработаны (спроектированы)
  - Остальные добавляются по готовности

Интеграционное тестирование, в отличие от модульного, проверяет не корректную функциональность модулей, а их взаимодействие между собой. Тесты очень похожи на модульные, но имеют другую направленность. Внутри могут тестироваться:

- Протоколы, взаимодействие между объектами, проверяется что протокол идет именно в том порядке, в котором необходимо.
- Пользовательские графические интерфейсы и другое

Есть несколько рабочих модулей и проверяется как они взаимодействуют между собой В модульных тестах будет набор вызовов, который будет ответственен за функционал одного модуля, а в интеграционном будут такие входные данные, которые пройдут целиком через всю цепочку программных модулей.

Может начинать разрабатываться когда разработана уже минимум 2 компонента. Остальные модули можно добавить по готовности.

Последовательность тестов обычно такая:

- Модульное тестирование
- Интеграционное тестирование
- Системное тестирование
- Приемочное тестирование

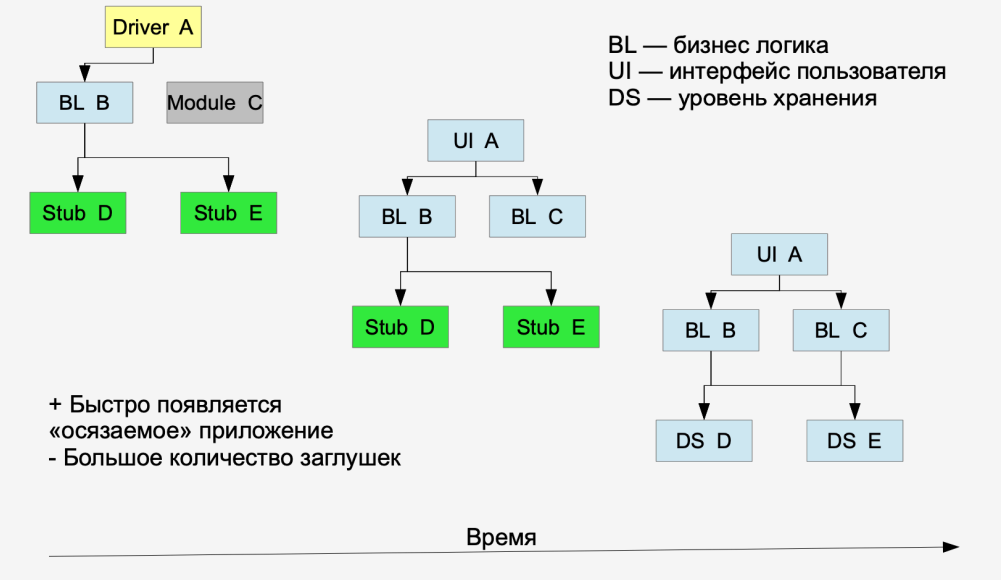
2. Алгоритм интеграционного тестирования.

Для алгоритма интеграционного тестирования можно выделить следующие этапы:

- Определение стратегии интеграции (из 5 имеющихся)
- Выбор модулей для интеграции
- Разработка тестовых сценариев
- Настройка и выполнение тестов
- Анализ/отладка ошибок
- Расширение тестового покрытия при необходимости

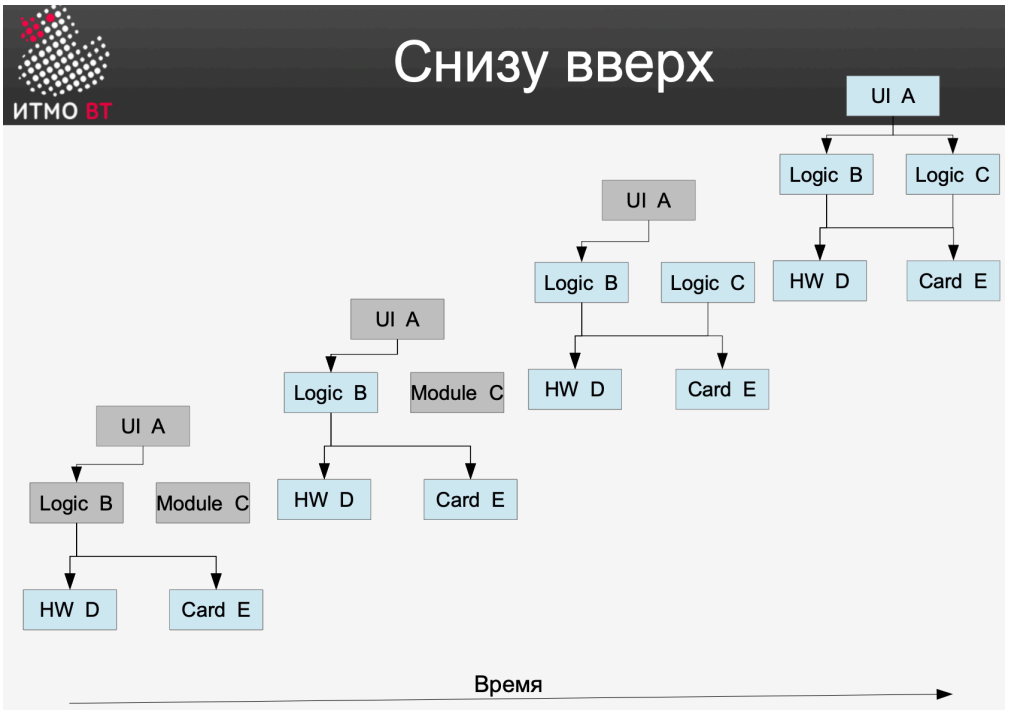
Стратегии интеграции:

1. **Сверху-вниз**



Сначала тестируют все высокоуровневые модули, затем постепенно добавляют низкоуровневые

2. **Снизу-вверх**



Сначала тестируют все низкоуровневые модули, процедуры или функции. Затем собирают следующий уровень модулей и проводят интеграционное тестирование

3. **Функциональная (end-to-end)**

По одной функциональности. Собрали 1 сценарий UI-Логика-БД, добавили еще один такой-же

4. **Ядро (backbone)**

Экран, клавиатура, мышь работают с минимальными функционалом
Добавить цвета на экран, колесо прокрутки …

Добавляется основное ядро системы с минимальным функционалом, затем добавляется дополнительный функционал по одному

5. **Большой взрыв (big bang)**

Собрать все вместе и молиться

3. Концепции и подходы, используемые при реализации интеграционного тестирования

- Интегрировать в первую очередь наиболее сложные компоненты
  - Снизит риски
  - Больше времени для исправления ошибок
- Выбирать порядок интеграции с учетом порядка разработки
- Использовать регрессионное тестирование после каждого этапа интеграции
- Автоматизировать тесты

1. Чем сложнее модули и взаимодействие между ними, тем больше времени потратится на тестирование. Поэтому сначала интегрируются наиболее сложные модули, а затем добавляются модули проще.

- Это оставит больше времени для исправления ошибок при интеграции компонентов
- И риски снизятся, например, что ПО не успеет подготовиться в срок

2. Порядок интеграции нужно сопоставлять с порядком разработки. Например, если есть два сложных в интеграции модуля, то их нужно поставить на первое место интеграции
3. Использовать регрессионные тесты - если добавился новый функционал, то прогоняются старые тесты, а потом добавляются новые. Это позволяет все время контролировать, что внесение нового модуля не испортит систему.
4. Автоматизировать тесты.