1. Термин "Three Amigos session" обычно относится к совместной встрече или обсуждению, в которых участвуют три человека, представляющих разные роли или точки зрения в рамках проекта или процесса разработки. Цель такой встречи заключается в том, чтобы способствовать коммуникации, согласованию и общему пониманию между этими тремя ключевыми ролями.

Три роли, которые часто связаны с понятием "Three Amigos", включают:

1. Бизнес-аналитик или Владелец продукта: Этот человек представляет бизнес-перспективу и отвечает за определение требований, функций и возможностей, которые должны быть реализованы.
2. Разработчик: Разработчик отвечает за реализацию технических аспектов проекта на основе предоставленных бизнес-аналитиком или Владельцем продукта требований.
3. Тестировщик или Аналитик по качеству (QA): Роль тестировщика заключается в обеспечении качества и функциональности продукта путем выявления и сообщения о дефектах и проблемах во время тестирования.

В ходе сессии "Three Amigos" эти три роли объединяются для обзора и обсуждения пользовательских историй, функций, требований или любого другого аспекта проекта, который требует сотрудничества между бизнесом, разработкой и тестированием. Сессия направлена на выявление потенциальных пробелов, разъяснение неясностей и обеспечение общего понимания работы, которую необходимо выполнить. Она помогает предотвратить недоразумения, уменьшить риск дефектов и повысить сотрудничество между членами команды.

Термин "Three Amigos" подчеркивает важность сотрудничества и коммуникации между этими тремя ролями для достижения успешного результата в проекте или разработке. Сам термин происходит от идеи трех близких друзей или коллабораторов, работающих вместе для достижения общей цели.

2. Да, Test-Driven Development (TDD) и Behavior-Driven Development (BDD) могут успешно использоваться вместе и часто дополняют друг друга в методологии разработки программного обеспечения. TDD и BDD - это подходы, акцентирующие внимание на тестировании и обеспечении качества программного обеспечения, но они имеют немного разные акценты и уровни абстракции.

Test-Driven Development (TDD): TDD - это практика разработки, при которой разработчики пишут тесты для определенной функциональности до написания фактического кода для реализации этой функциональности. Процесс TDD обычно следует этим шагам:

1. Написание теста, который не проходит и указывает на желаемое поведение или функциональность.
2. Написание минимально необходимого кода для прохождения теста.
3. Рефакторинг кода для улучшения его дизайна, сохраняя при этом прохождение теста.

TDD более сосредоточено на написании юнит-тестов, которые проверяют поведение отдельных частей кода (таких как функции или методы). Эти тесты помогают разработчикам убедиться, что их код соответствует ожидаемым требованиям и работает корректно в изоляции.

Behavior-Driven Development (BDD): BDD - это расширение TDD, которое акцентирует более сильное взаимодействие между разработчиками, тестировщиками и бизнес-заказчиками. BDD смещает акцент с тестирования отдельных частей кода на тестирование поведения всего программного обеспечения в целом. BDD включает написание тестов в более понятном для человека формате, описывающем ожидаемое поведение программного обеспечения с точки зрения различных заинтересованных сторон.

Обычно BDD включает использование специализированного фреймворка, который позволяет писать тесты в естественном языке. Эти тесты часто называют "сценариями" и структурированы с помощью клавул Given-When-Then, чтобы описать контекст, действия и ожидаемые результаты.

Использование TDD и BDD вместе: TDD и BDD могут использоваться совместно для обеспечения всестороннего подхода к тестированию и обеспечению качества. Сценарии BDD могут служить более высокоуровневым представлением поведения, которое должно быть реализовано, и TDD может быть использован для реализации базовой функциональности, обеспечивая ее правильность с помощью юнит-тестов.

Сценарии BDD также могут помочь улучшить коммуникацию между техническими и нетехническими членами команды, обеспечивая общее понимание желаемого поведения. TDD, с другой стороны, обеспечивает поддержание кодовой базы с хорошими характеристиками поддерживаемости, расширяемости и хорошим уровнем тестирования на уровне юнита.

В итоге, TDD и BDD могут дополнять друг друга в разработке программного обеспечения, акцентируя внимание как на детализированной реализации отдельных частей (TDD), так и на общем поведении программы (BDD). Интеграция этих методологий может привести к более надежному и хорошо протестированному программному продукту.

3. Gherkin - это человеко-читаемый, структурированный язык, который используется для написания спецификаций и критериев приемки для программных приложений. Он часто применяется в разработке на основе поведения (Behavior-Driven Development, BDD) для определения того, как система программного обеспечения должна вести себя с точки зрения пользователя. Gherkin выступает в роли связующего звена между не-техническими заинтересованными сторонами (например, бизнес-аналитиками, владельцами продукта и экспертами в предметной области) и техническими членами команды (разработчиками и тестировщиками).

Gherkin использует простой синтаксис с ключевыми словами, которые описывают поведение программного обеспечения в формате, который легко понимать и обсуждать. Основная цель Gherkin - это определение выполнимых спецификаций, которые могут быть использованы как основа для автоматизированных тестов и как справочник для разработки.

Основные особенности языка Gherkin включают:

1. Синтаксис Given-When-Then: Сценарии Gherkin структурированы с использованием синтаксиса Given-When-Then. Каждый сценарий состоит из трех основных разделов:
   * Given: Начальный контекст или состояние системы.
   * When: Действие или событие, которое происходит.
   * Then: Ожидаемый результат или исход.
2. Ключевые слова: Gherkin использует конкретные ключевые слова для определения разных разделов сценария, таких как "Given", "When", "Then", "And" и "But". Эти ключевые слова делают сценарии легко читаемыми и понятными.
3. Образцы сценариев: Gherkin поддерживает образцы сценариев, которые позволяют создавать шаблонные сценарии с заполнителями для конкретных значений. Это помогает создавать несколько сценариев с разными входными данными.
4. Метки: Сценарии могут быть помечены метками для группировки связанных сценариев и более удобной организации тестов.
5. Комментарии: Gherkin поддерживает однострочные и многострочные комментарии для предоставления дополнительного контекста или пояснений.
6. Локализация: Gherkin можно писать на нескольких языках, что делает его доступным для команд, работающих в разных регионах.
7. Интеграция с инструментами: Сценарии Gherkin могут быть преобразованы в выполнимые тесты с использованием различных фреймворков для тестирования BDD, таких как Cucumber для Java, Behave для Python, SpecFlow для .NET и другие.

Пример сценария на Gherkin:

gherkinCopy code

Функция: Корзина для покупок

Как клиент

Я хочу добавлять товары в свою корзину для покупок

Чтобы затем приобрести их

Сценарий: Добавление товара в корзину для покупок

Дано, что клиент находится на странице товара

Когда клиент добавляет "Товар А" в корзину

Тогда корзина должна содержать "Товар А"

И итоговая сумма корзины должна быть $X

В этом примере Gherkin описывает сценарий, в котором клиент добавляет товар в свою корзину для покупок. Синтаксис Given-When-Then явно определяет начальное состояние, действие и ожидаемые результаты сценария.

Gherkin помогает согласовывать технических и нетехнических членов команды, предоставляя общий язык для обсуждения и документирования поведения программного обеспечения.

Конечно, вот еще несколько примеров сценариев на языке Gherkin:

Пример 1: Вход в систему

gherkinCopy code

Функция: Вход в систему

Как пользователь

Я хочу войти в систему

Чтобы получить доступ к своему аккаунту

Сценарий: Успешный вход в систему

Дано, что я нахожусь на странице входа

Когда я ввожу свои учетные данные

И я нажимаю кнопку "Войти"

Тогда я должен видеть страницу своего аккаунта

Пример 2: Покупка товара

gherkinCopy code

Функция: Покупка товара

Как покупатель

Я хочу приобрести товары

Чтобы совершить покупку

Сценарий: Добавление товаров в корзину и оформление заказа

Дано, что я нахожусь на странице с продуктами

Когда я добавляю "Товар А" и "Товар В" в корзину

И я перехожу в корзину

Тогда я должен видеть "Товар А" и "Товар В" в корзине

И итоговая сумма заказа должна быть верной

Когда я нажимаю кнопку "Оформить заказ"

Тогда я должен видеть подтверждение о заказе

Пример 3: Создание задачи в управлении проектами

gherkinCopy code

Функция: Создание задачи

Как член команды проекта

Я хочу создать новую задачу

Чтобы организовать работу

Сценарий: Создание новой задачи

Дано, что я нахожусь на доске задач

Когда я нажимаю кнопку "Создать задачу"

И я ввожу название "Новая задача"

И я выбираю приоритет "Высокий"

И я указываю описание задачи

И я нажимаю кнопку "Создать"

Тогда я должен видеть задачу "Новая задача" на доске

Эти примеры демонстрируют, как сценарии на языке Gherkin описывают разные функциональности и поведения программного обеспечения. Каждый сценарий включает в себя разделы Given, When и Then, которые описывают начальное состояние, действие и ожидаемые результаты.