Сравнение подходов к разработке

📊 Старый подход vs Новый TDD-focused подход

Аспект	Старый подход ❌	Новый подход ✅	Результат
Фокус	Метрики покрытия (10%, 15%, 25%)	Качество кода и дизайна	Надежность вместо цифр
Количество тестов	200-300 за спринт	16-20 за спринт	Качество > Количество
Скорость создания	10 тестов/минуту	1 тест за 25-30 минут	Осмысленные тесты
Процесс	Test-Last (тесты после кода)	Test-First (TDD)	Лучший дизайн
Запуск тестов	В конце спринта	После каждого изменения	Мгновенная обратная связь
Красные тесты	Игнорировались	Блокируют коммиты	100% зеленые всегда

🚨 Ключевые проблемы старого подхода

1. Массовое создание тестов

```
Sprint 33:
  - День 155: 201 тест за 20 минут
  - День 156: 100 тестов за 15 минут
  Проблема: Тесты без привязки к коду, формальный подход
```

2. Фокус на метриках

```
Цели спринтов:
  - Sprint 33: "Достичь 15% покрытия"
  - Sprint 34: "Достичь 10% покрытия"
  - Sprint 38: "Достичь 15% покрытия"
  Проблема: Покрытие стало целью, а не побочным эффектом
```

3. Отложенный запуск

```
Паттерн:
  1. Создать 200+ тестов
  2. Потом исправлять компиляцию
  3. Потом запускать
```

```
  4. Потом чинить падающие
  Проблема: Накопление технического долга
```

✅ Преимущества нового подхода

1. Red-Green-Refactor цикл

```
Каждая функция:
  1. Написать тест (5 минут)
  2. Запустить - увидеть красный (1 минута)
  3. Написать минимальный код (10 минут)
  4. Запустить - увидеть зеленый (1 минута)
  5. Рефакторинг (5-10 минут)
  Итого: 25-30 минут на полный цикл
```

2. Качественные метрики

```
Вместо:
  - Количество тестов: 301
  - Покрытие: 15%

Теперь:
  - TDD Compliance: 100%
  - Test Stability: 100%
  - Refactoring Rate: 60%
```

3. Защитные механизмы

```
Автоматизация:
  - Pre-commit hooks блокируют красные тесты
  - CI/CD отклоняет PR с падающими тестами
  - Ежедневные отчеты о соблюдении TDD
```

📈 Прогноз результатов

Старый подход (факт)

- **Sprint 33**: 301 тест → 5.60% покрытия
- **Sprint 34**: 258 тестов → ~7% покрытия
- **Sprint 38**: 250+ тестов → 17.22% покрытия
- **Проблемы**: Тесты не запускались, накапливался техдолг

Новый подход (прогноз)

- **Sprint 39**: 16 тестов → +2-3% покрытия органически
- **Sprint 40**: 20 тестов → +3-4% покрытия органически
- **Sprint 45**: Достижение 40-50% покрытия естественным путем
- **Преимущества**: 100% зеленые тесты, уверенность в коде

🎯 Измеримые цели

Неделя 1 (Sprint 39)

- [ ] Внедрить pre-commit hooks
- [ ] Настроить CI/CD с обязательными тестами
- [ ] Провести 16 полных TDD циклов
- [ ] 100% тестов зеленые каждый день

Месяц 1

- [ ] TDD Compliance Rate > 95%
- [ ] Среднее время цикла < 30 минут
- [ ] 0 красных тестов в main ветке
- [ ] Снижение багов на 50%

Квартал 1

- [ ] Покрытие кода 40-50% (органически)
- [ ] Время на исправление багов -80%
- [ ] Скорость разработки +30%
- [ ] Удовлетворенность команды +50%

💡 Культурный сдвиг

Было:

- "Нужно написать 200 тестов для покрытия"
- "Сначала код, потом тесты"
- "Красные тесты исправим потом"
- "Главное - метрики для отчета"

Стало:

- "Нужно написать тест для этой функции"
- "Без теста нет кода"
- "Красный тест = стоп работа"
- "Главное - уверенность в коде"

🔄 Переходный период

Sprint 39 (текущий)

1. Остановить гонку за покрытием
2. Внедрить TDD инфраструктуру

3. Обучить команду новому процессу
4. Начать с малого - 3-5 циклов в день

Sprint 40-42

1. Увеличить количество циклов до 5-7
2. Сократить время цикла до 20-25 минут
3. Автоматизировать все рутинные проверки
4. Измерять качественные метрики

Sprint 43+

1. TDD становится естественным процессом
2. Покрытие растет органически
3. Баги находятся на этапе разработки
4. Рефакторинг без страха

📊 ROI нового подхода

Краткосрочный (1-2 спринта)

- **Инвестиции**: -50% скорость написания кода
- **Возврат**: -80% времени на исправление багов
- **Итого**: +30% общей производительности

Среднесрочный (3-6 спринтов)

- **Инвестиции**: Время на обучение TDD
- **Возврат**: +50% скорость разработки
- **Итого**: +100% уверенности в коде

Долгосрочный (6+ спринтов)

- **Инвестиции**: Культурные изменения
- **Возврат**: -90% критических багов в production
- **Итого**: x10 удовлетворенность пользователей

Вывод: Переход от количественных метрик к качественному TDD процессу требует изменения мышления, но дает кратно лучшие результаты в долгосрочной перспективе.