ЛЕКЦИЯ 12

Диагностика и устранение неполадок

CSE842 Capstone Project
ОП 6B06102 Computer Science



План лекции

1. Диагностика
2. Отчёт
3. Повтор
4. Гипотезы
5. Журналы
6. bisect
7. Исправление
8. Регрессия

Диагностика по Google SRE



Наблюдать

симптомы и сигналы



Предположить

гипотеза о причине



Проверить

эксперимент, не спор

Ловушки: нерелевантные симптомы, совпадение вместо причины

Отчёт о дефекте



Заголовок

смежная бронь отклоняется



Шаги

версия, окружение, данные



Ожидание и факт

201 против 409



Серьёзность

и приоритет — разные

Серьёзность и приоритет

Приоритет

Срочно, но мелко опечатка на главной перед защитой	**Немедленно** двойная бронь, падение API
Позже выравнивание в отчёте	**Важно, не срочно** редкий сбой годового отчёта

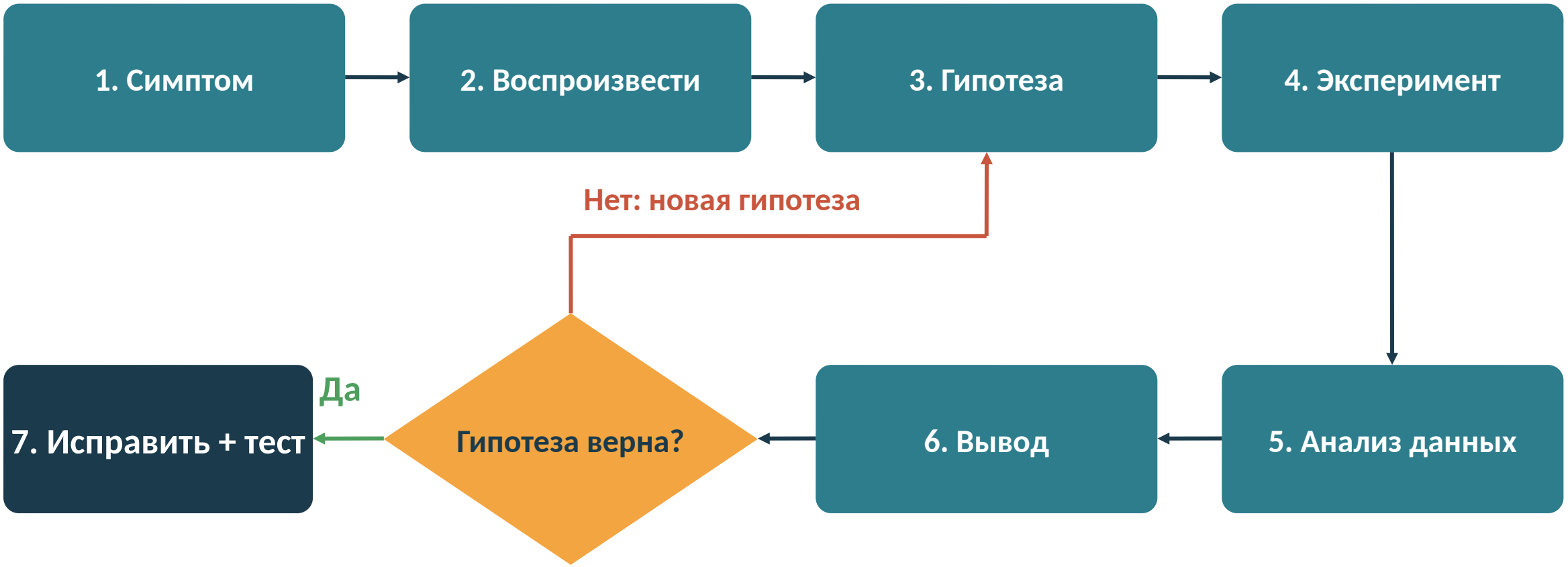
Серьёзность

Воспроизвести и минимизировать

Симптом	Условия	Минимум	Тест
«иногда отклоняет»	данные, порядок, время	одна функция, два интервала	падает до исправления

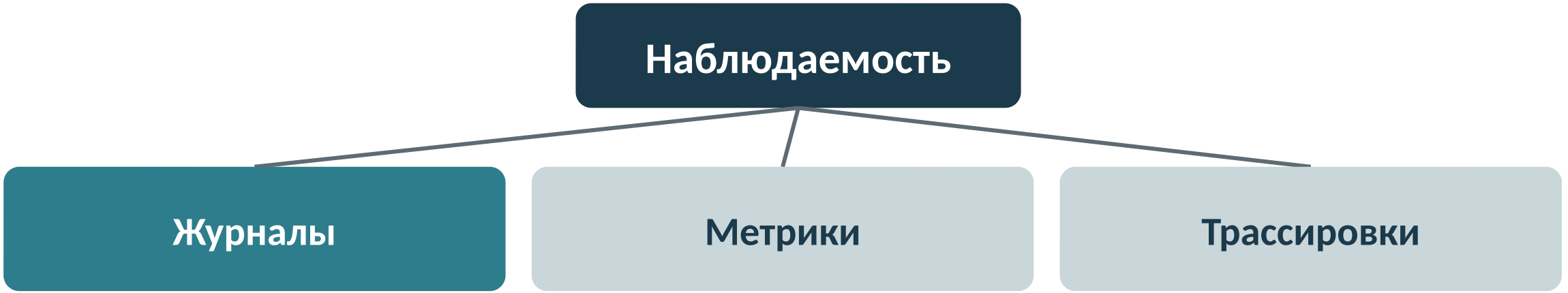
Delta debugging: удалять части входа, пока сбой сохраняется

Цикл диагностики



Эксперимент — журнал, отладчик, имитация, сравнение версий

Три сигнала наблюдаемости



Никогда не журналировать пароли и токены

Уровни журнала

Уровень	Пример
DEBUG	проверка пересечения eq=3, найдено 0
INFO	booking.created id=17 req=7f2a
WARNING	outbox: попытка 3 отправки
ERROR	SMTP недоступен, событие отложено
CRITICAL	БД недоступна, API → 503

Воспроизведение BUG-14

l12_debug.py

```
import logging
import sys
logging.basicConfig(stream=sys.stdout, format="%(levelname)s %(message)s",
                    level=logging.DEBUG)
log = logging.getLogger("booking")

def overlaps_v1(a, b):                    # дефект BUG-14
    return a[0] <= b[1] and b[0] <= a[1]
def overlaps_v2(a, b):                    # исправление
    return a[0] < b[1] and b[0] < a[1]

case = ((10, 12), (12, 14))               # шаг воспроизведения
for f in (overlaps_v1, overlaps_v2):
    res = f(*case)
    log.debug("%s%s -> %s", f.__name__, case, res)
    if res: log.error("ложный конфликт: смежные слоты")
regress = [((10, 12), (11, 13), True), ((10, 12), (12, 14), False),
           ((8, 9), (10, 12), False)]
ok = all(overlaps_v2(a, b) == exp for a, b, exp in regress)
log.info("регрессионный набор: %s", "PASS" if ok else "FAIL")
```

```
$ python l12_debug.py
DEBUG overlaps_v1((10, 12), (12, 14)) -> True
ERROR ложный конфликт: смежные слоты
DEBUG overlaps_v2((10, 12), (12, 14)) -> False
INFO регрессионный набор: PASS
```

- v1: True — ложный конфликт
- v2: False
- регрессия PASS

[s, e] вместо полуинтервала [s, e)

git bisect: бинарный поиск

100

коммитов между good и bad

≈ 7

проверок: $\log_2 100$

1

коммит с дефектом

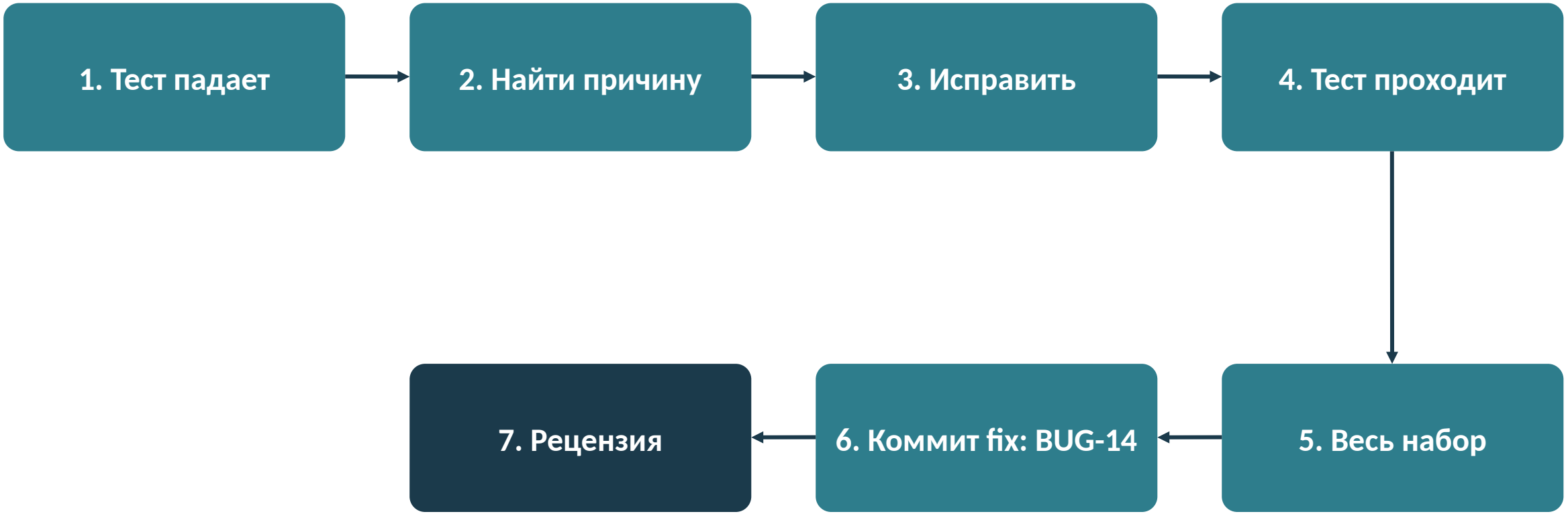


git bisect start; git bisect bad; git bisect good v0.9; git bisect run тест

Классы дефектов

Класс	Пример	Обнаружение
граница ±1	<= вместо <	граничные тесты
время	+05:00 и Z как строки	тесты поясов
гонка	две брони слота	многопоточный тест
контракт	id и booking_id	контрактные тесты

Порядок исправления дефекта

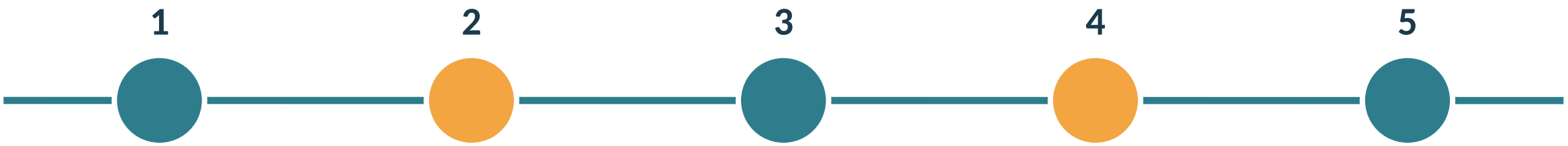


Не «закомментировать тест» и не try/except: pass

Журнал дефектов

ID	Симптом	Причина	Регрессия
BUG-14	смежная бронь	<= вместо <	31/31
BUG-17	два письма	нет проверки id	33/33
BUG-19	500 на пустом теле	нет обработки JSON	34/34

«Пять почему» для BUG-17



1	2	3	4	5
письмо ушло дважды	событие повторено	пометка после отправки	нет проверки id	контракт без рецензии

Blameless postmortem: улучшаем процесс, а не ищем виноватых

Вопросы для самопроверки

1. Отчёт о дефекте
2. Минимизация
3. Цикл SRE
4. Сигналы
5. git bisect
6. Классы дефектов
7. Порядок исправления
8. Регрессия



Неделя 12: журнал дефектов и регрессии (3 балла)