**1. Определение и основные цели**

**Измерительные методы оценки программ** — это подходы к количественному анализу характеристик программного обеспечения с использованием метрик. Они позволяют объективно оценить:

**Размер кода** (объем, сложность).

**Качество** (надежность, сопровождаемость).

**Производительность** (время выполнения, использование ресурсов).

**Основные цели применения:**

**Контроль качества** ПО на всех этапах разработки.

**Выявление проблемных мест** (сложный код, низкое покрытие тестами).

**Сравнение альтернативных решений** (оптимизация алгоритмов).

**Прогнозирование трудозатрат** на поддержку и развитие проекта.

**2. Примеры метрик и их назначение**

| **Тип метрик** | **Примеры** | **Для чего используются** |
| --- | --- | --- |
| **Метрики размера** | Количество строк кода (SLOC) | Оценка объема кода, трудозатрат на разработку и поддержку. |
|  | Количество функций/методов | Анализ модульности и структурированности кода. |
| **Метрики сложности** | Цикломатическая сложность (Маккейб) | Определение числа независимых путей выполнения кода. Высокие значения = риск ошибок. |
|  | Глубина вложенности | Оценка читаемости: чем больше вложенность, тем сложнее поддерживать код. |
| **Метрики качества** | Коэффициент сопровождения (MI) | Расчет индекса от 0 до 100: чем выше, тем проще сопровождать код. Формула: MI = 171 - 5.2\*ln(Halstead Volume) - 0.23\*(Cyclomatic Complexity) - 16.2\*ln(LOC) |
|  | Покрытие тестами (%) | Определение доли кода, проверенной автоматическими тестами. Цель: 70–90%. |

**3. Условия применения и ограничения**

**Когда применяются метрики?**

**На этапе разработки**: для контроля сложности кода (например, проверка цикломатической сложности в CI/CD).

**При рефакторинге**: выявление "дурно пахнущего" кода (god objects, spaghetti code).

**Для аудита безопасности**: поиск уязвимостей через статический анализ (например, высокое число SQL-инъекций в метриках уязвимостей).

**Ограничения измерительных методов:**

**Ложные срабатывания**. Некоторые метрики (например, SLOC) не учитывают контекст: 100 строк чистого кода ≠ 100 строк "спагетти".

**Неполнота данных**. Метрики не всегда отражают бизнес-логику или UX-качество.

**Зависимость от инструментов**. Разные анализаторы могут давать противоречивые результаты (например, ESLint vs. Pylint).

**Субъективная интерпретация**. Высокая цикломатическая сложность может быть оправдана в алгоритмически сложных модулях.