**Метрики для оценки эффективности процесса разработки проекта “Рецептовед”**

1. **Метрика: Производительность труда (Productivity)**
   * **Формула:**

Productivity = LOC / Рабочее время, затраченное на проект

* + **Единица измерения:** LOC / человеко-час
  + **Цель:** Повышение производительности труда разработчиков.
  + **Описание:** Метрика позволяет оценить, сколько строк кода (LOC) создаётся за единицу времени. Это помогает понять, насколько эффективно используется рабочее время разработчиков.
  + **Пример:** Если за 100 человеко-часов было написано 5000 строк кода, то производительность составит 50 LOC/человеко-час.

1. **Метрика: Эффективность обнаружения ошибок (Phase Containment Effectiveness, PCE)**
   * **Формула:**

PCE = [D(I, I) / Sum(D(I, J))]\*100%

Где:

* + - *D*(*I*,*J*) — количество ошибок, внесённых на фазе I и обнаруженных на фазе J.
    - I, J — номера фаз проекта (1 — планирование, 2 — разработка требований, 3 — разработка дизайна, 4 — кодирование, 5 — тестирование).
  + **Единица измерения:** %
  + **Цель:** Повышение качества разрабатываемого ПО за счёт раннего обнаружения ошибок.
  + **Описание:** Метрика показывает, насколько эффективно ошибки обнаруживаются на той же фазе, на которой они были внесены. Чем выше значение PCE, тем меньше ошибок переходит на следующие этапы разработки.
  + **Пример:** Если на фазе кодирования было внесено 10 ошибок, и 8 из них были обнаружены на этой же фазе, то PCE для фазы кодирования составит 80%.

**Метрики для оценки качества программного продукта**

1. **Метрика: Плотность ошибок в продукте (In Process Faults, IPF)**
   * **Формула:**

IPF = Число обнаруженных ошибок до выпуска релиза */* LOC

* + **Единица измерения:** Ошибка / LOC
  + **Цель:** Повышение качества разрабатываемого ПО за счёт снижения количества ошибок в готовом продукте.
  + **Описание:** Метрика показывает, сколько ошибок содержится в каждой тысяче строк кода. Чем ниже значение IPF, тем выше качество продукта.
  + **Пример:** Если в продукте из 10 000 строк кода было обнаружено 20 ошибок до выпуска, то плотность ошибок составит 2 ошибки на 1000 строк кода.

1. **Метрика: Плотность ошибок на этапе тестирования (Product Fault Density, PFD-Test)**
   * **Формула:**

PFD-Test = Число ошибок, допущенных на этапе тестирования / Количество тестов

* + **Единица измерения:** Ошибка / тест
  + **Цель:** Повышение качества тестирования и снижение количества ошибок, обнаруженных на этапе тестирования.
  + **Описание:** Метрика позволяет оценить, сколько ошибок приходится на каждый тест. Чем ниже значение PFD-Test, тем эффективнее процесс тестирования.
  + **Пример:** Если в процессе тестирования было проведено 100 тестов и обнаружено 5 ошибок, то плотность ошибок на этапе тестирования составит 0.05 ошибки на тест.

**Чек-лист для проверки корректности кода перед инспекцией**

1. **Проверка инициализации всех локальных переменных.**

Убедитесь, что все переменные инициализированы перед использованием.

1. **Проверка отсутствия макросов для переименования операторов и ключевых слов.**

Убедитесь, что в коде нет макросов, переименовывающих операторы или ключевые слова.

1. **Проверка использования перечислимых типов (enum) вместо препроцессорных констант.**

Убедитесь, что для констант используются перечислимые типы, а не макросы.

1. **Проверка отсутствия глобальных идентификаторов, начинающихся с подчерка или содержащих \_\_.**

Убедитесь, что в коде нет таких идентификаторов.

1. **Проверка длины строк кода (не более 100 символов).**

Убедитесь, что длина строк не превышает 100 символов.

1. **Проверка наличия комментариев в конце операторов ветвлений и циклов, если тело содержит более 5 строк.**

Убедитесь, что такие комментарии присутствуют.

1. **Проверка использования директив условной компиляции вместо комментирования блоков кода.**

Убедитесь, что для исключения кода из компиляции используются директивы условной компиляции.

1. **Проверка отсутствия оператора goto.**

Убедитесь, что оператор goto не используется, за исключением крайних случаев.

1. **Проверка наличия заголовков файлов и функций в виде комментариев установленного образца.**

Убедитесь, что каждый файл и функция имеют соответствующие заголовки.

1. **Проверка использования констант в условиях на первом месте.**

Убедитесь, что в условиях константы стоят на первом месте, например:

if (0 == j) { ... }