Пара №8

Теоретическая часть

1. Метрики ПО: Назначение и Применение

1) Метрика программного обеспечения — это количественная мера, используемая для оценки различных аспектов программного обеспечения, таких как его качество, сложность, производительность и эффективность. Метрики помогают разработчикам и менеджерам проектов принимать обоснованные решения на основе данных.

2) Оценка сложности - метрики сложности, такие как цикломатическая сложность, помогают оценить, насколько сложен код для понимания и поддержки. Это позволяет выявить участки кода, которые могут потребовать рефакторинга или дополнительного тестирования.

Контроль качества - метрики качества, такие как количество дефектов на тысячу строк кода, помогают оценить качество программного обеспечения. Это позволяет командам сосредоточиться на улучшении проблемных областей и повышении общего качества продукта.

Улучшение стиля кода - метрики стиля кода, такие как соблюдение стандартов кодирования и соглашений об именовании, помогают поддерживать единообразие и читаемость кода. Это облегчает поддержку и развитие проекта.

3) При ревью кода метрики могут быть использованы для выявления потенциальных проблем и улучшения качества кода.

Сложность кода - метрики сложности могут указать на участки кода, которые слишком сложны и требуют упрощения. Это помогает ревьюерам сосредоточиться на наиболее проблемных частях кода.

Покрытие тестами - метрики покрытия тестами показывают, какие части кода недостаточно протестированы. Это помогает ревьюерам выявить участки, требующие дополнительного тестирования.

Соблюдение стандартов кодирования - метрики могут показать, насколько код соответствует установленным стандартам и соглашениям. Это помогает поддерживать единообразие и читаемость кода в проекте.

2. Метрики Сложности

1)Измерять сложность кода нужно по нескольким причинам:

*Обнаружение проблемных участков* - высокосложный код труднее понимать, тестировать и поддерживать. Измерение сложности помогает выявить такие участки кода, которые могут потребовать дополнительного внимания.

*Улучшение качества кода* - снижение сложности кода способствует улучшению его качества, делая его более читаемым и понятным для других разработчиков. Это облегчает поддержку и развитие проекта.

*Снижение количества ошибок* - сложный код чаще содержит ошибки. Измерение сложности помогает выявить участки кода, которые могут быть более подвержены ошибкам, и сосредоточиться на их улучшении.

*Повышение эффективности разработки* - менее сложный код легче изменять и расширять, что способствует повышению общей эффективности разработки и снижению затрат на поддержку

Код высокой сложности вызывает такие проблемы как трудность в понимании повышенную вероятность ошибок затруднения при тестировании и снижение производительности

2) *Количество параметров функции* - эта метрика измеряет количество параметров, которые принимает функция. Большое количество параметров может указывать на сложную и трудную для понимания функцию, которая может требовать рефакторинга.

*Цикломатическая сложность* - Эта метрика измеряет количество независимых путей выполнения в коде. Высокая цикломатическая сложность указывает на сложный код, который труднее тестировать и поддерживать.

3. Метрики Стилистики (Качества Кода)

1) Метрики стилистики используются для оценки и улучшения стиля написания кода. Они помогают обеспечить единообразие и читаемость кода, что является ключевым для его сопровождаемости и поддержки.

2) Длина идентификаторов - измеряет среднюю и минимальную длину имен переменных, функций и других идентификаторов. Короткие имена могут быть менее понятными, в то время как слишком длинные имена могут затруднять чтение кода.

Количество комментариев - измеряет отношение количества строк комментариев к общему количеству строк кода. Комментарии помогают объяснить сложные участки кода, но их избыток может указывать на плохую читаемость кода.

Длина строки кода - измеряет максимальную и среднюю длину строк кода. Слишком длинные строки могут быть трудными для чтения и понимания, особенно на экранах с ограниченным пространством.

3) Слишком короткие имена переменных, такие как x или tmp, считаются плохой практикой т.к недостаточно информативны, бывают трудности в понимании, а также есть повышенная вероятность конфликтов

*Практическая часть (анализ кода)*

1. Метрики стилистики
2. Идентификаторы:

|  |  |
| --- | --- |
| Идентификатор | Оценка понятности (1-5) |
| requests | 5 |
| pd | 4 |
| plt | 4 |
| datetime | 5 |
| timedelta | 5 |
| ET | 3 |
| currencies | 5 |
| currency\_data | 5 |
| end\_date | 5 |
| start\_date | 5 |
| current\_date | 5 |
| formatted\_date | 5 |
| url | 5 |
| response | 5 |
| root | 4 |
| date\_record | 5 |
| valute | 4 |
| char\_code | 4 |
| value | 5 |
| nominal | 5 |
| normalized\_value | 5 |
| writer | 4 |
| df | 4 |
| currency | 5 |
| calc | 5 |
| a | 2 |
| b | 2 |
| op | 3 |
| r | 2 |
| find\_max | 5 |
| arr | 3 |
| m | 2 |
| x | 2 |

Средняя длина идентификаторов

|  |  |
| --- | --- |
| Функция | Средняя длина идентификаторов |
| Основной скрипт | 6.5 |
| calc | 1.5 |
| find\_max | 2.3 |

В основном скрипте средняя длина идентификаторов составила 6.5 символов, что указывает на использование достаточно длинных и понятных имен. В функциях calc и find\_max средняя длина идентификаторов ниже, что может указывать на менее понятные имена переменных.

Комментарии:

1)Общее количество строк кода: 100

Количество строк с комментариями: 20

Процент строк с комментариями= (20/100)\* 100= 20

2) Оценка - 4

Структура:

Общее количество строк кода: 100

Сумма символов во всех строках кода: 4950

Средняя длина строки кода = (4950 / 100)= 49.5

2) Нарушения:

1.url = f"[https://www.cbr.ru/scripts/XML\_daily.asp?date\_req={formatted\_date}"](https://www.cbr.ru/scripts/XML_daily.asp?date_req=%7Bformatted_date%7D%22)

Превышает рекомендуемую длину

2. import xml.etree.ElementTree as ET

# здесь будет пустая строка

# Список валют, которые мы будем анализировать (обновлен по заданию)

currencies = ['DKK', 'CAD', 'NZD', 'PLN']

Некоторые блоки кода могли бы использовать дополнительные пустые строки для улучшения читаемости.

2. Анализ и Рекомендации

1. Сильные стороны:

Хорошая структура и организация кода - он разделен на логические блоки, что облегчает его понимание и поддержку.

Использование комментариев - код содержит комментарии, которые объясняют основные шаги и назначение блоков кода. Это улучшает читаемость и понимание кода.

Слабые стороны:

Длина строк кода - некоторые строки кода превышают рекомендуемую длину в 79 символов, что может затруднять чтение и понимание кода.

Короткие имена переменных в функциях - в функциях calc и find\_max используются короткие имена переменных, такие как a, b, op, r, m, x, которые не несут достаточной информации о своем назначении.

2. Разбиение длинных строк кода.

url = ( "

[https://www.cbr.ru/scripts/XML\_daily.asp?"](https://www.cbr.ru/scripts/XML_daily.asp?%22)

f"date\_req={formatted\_date}"

)

Переименование переменных в функциях.

В функции find\_max можно использовать следующие имена :

def find\_max(array):

max\_value = array[0]

for element in array:

if element > max\_value:

max\_value = element

return max\_value