Приднестровский Государственный Университет им. Т.Г. Шевченко

Бендерский Политехнический Филиал

Кафедра «Промышленность и информационные технологии»

**Практическая работа №8**

По УП 03.01 «Учебная практика»

ТЕМА**:** «Использование метрик программного продукта»

Выполнил:

студент 3 курса, БК22АР52ИС2 группы,

дневной формы обучения, специальность:

«Информационные системы и программирование»

Урыта Эдуард Игоревич

Руководитель учебной практики:

Лазовский Александр Валерьевич

Бендеры, 2025 г.

**Использование метрик программного продукта.**

**Метрики программного продукта** — это количественные характеристики, которые позволяют оценивать качество, эффективность, производительность, надёжность и сложность ***ПО***, а также управлять разработкой и сопровождением.

**Они необходимы для**:

* Повышения качества кода и архитектуры;
* Эффективного управления проектами;
* Оптимизации производительности;
* Улучшения процессов тестирования;
* Повышения удовлетворённости пользователей.

**Основные категории метрик и их примеры.**

1. **Метрики кода *(Code Metrics).***

Используются для анализа **внутреннего качества и сложности кода**.

**Размер кода:**

* ***SLOC (Source Lines of Code)*** — общее количество строк в исходниках;
* Количество функций, классов, методов;
* *Средняя длина функции/метода* — влияет на читаемость и сопровождение.

**Сложность кода:**

* *Цикломатическая сложность* ***(McCabe)*** — сколько логических путей в коде:  
  *Чем выше, тем труднее тестировать и понимать*;
* *Когезия* ***(Cohesion)*** — насколько тесно связаны части модуля;
* *Связность* ***(Coupling)*** — насколько сильно модули зависят друг от друга:  
  *Высокая связность = трудно изменять один модуль без влияния на другие.*

**Качество кода:**

* *Дублирование кода* — повторяющиеся участки, которые следует выносить;
* *Плотность дефектов* — сколько ошибок на ***1000*** строк;
* *Покрытие стандартов* — соответствие стилю кодирования, *через* ***Lint****,* ***SonarQube*** *и др*.

1. **Метрики процесса разработки *(Process Metrics).***

Позволяют следить за **прогрессом и эффективностью** команды.

**Временные метрики:**

* ***Lead Time*** — время от запроса до поставки;
* ***Cycle Time*** — время выполнения конкретной задачи;
* ***Time to Fix Bugs*** — сколько времени уходит на устранение ошибки.

**Производственные метрики:**

* ***Velocity*** — объём задач, выполненных за спринт;
* ***Deployment Frequency*** — как часто вы выкатываете обновления;
* ***Change Failure Rate*** — процент релизов, в которых были обнаружены ошибки.

**Качество процессов:**

* Количество ***reopened*** задач *(повторно открытых)*;
* Количество отменённых задач — может указывать на плохую планировку.

1. **Метрики производительности приложения *(Performance Metrics).***

Используются для оценки **работоспособности *ПО* в условиях эксплуатации**.

* **Время отклика** ***(Response Time)*** — за сколько система отвечает пользователю.
* **Пропускная способность** ***(Throughput)*** — сколько операций в секунду.
* **Использование ресурсов**:
  + ***CPU***;
  + ***RAM***;
  + Сеть;
  + Диск.
* **Время простоя** ***(Downtime)*** — сколько времени система была недоступна.

1. **Метрики тестирования *(Testing Metrics).***

Помогают понять, **насколько хорошо протестирован продукт**.

* ***Code Coverage*** — процент кода, который проверяется автоматическими тестами.
* ***Test Pass Rate*** — соотношение успешных и проваленных тестов.
* ***Defect Density*** — число багов на объем кода.
* ***Defect Removal Efficiency (DRE)*** — какой процент найденных дефектов устраняется до релиза.
* ***Test Case Effectiveness*** — насколько часто тесты реально находят баги.

1. **Метрики пользовательского опыта *(UX / Usability Metrics).***

Измеряют **удовлетворённость и поведение пользователя**:

* Время загрузки *интерфейса/страницы*;
* Количество переходов до достижения цели *(минимизация* ***"фрикции"****)*;
* Поведенческие метрики:
  + Время в системе;
  + Показатель отказов;
  + Повторные визиты.
* Количество жалоб пользователей;
* ***Churn Rate*** — сколько пользователей покинули продукт за период.

**Как внедрять и использовать метрики.**

1. **Определите цели**:  
   Улучшение кода, оптимизация производительности, контроль прогресса и *т.д.*
2. **Выберите ключевые метрики** ***(KPI)***:  
   *Не перегружайте систему!* Лучше ***3–5*** ключевых показателей, чем ***20*** ненужных.
3. **Автоматизируйте сбор**:  
   Используйте ***CI/CD***, статический анализ кода, системы мониторинга, боты в ***GitHub***.
4. **Анализируйте регулярно**:  
   Встраивайте отчёты в ежедневные стендапы или спринт-ревью.
5. **Используйте данные для принятия решений**:  
   Если сложность растёт — *начните рефакторинг*. Если увеличиваются ошибки — *пересмотрите процесс тестирования*.

**Инструменты и сервисы.**

| **Категория:** | **Инструменты:** |
| --- | --- |
| Анализ кода | ***SonarQube, CodeClimate, ESLint, PMD, Pylint*** |
| Тестирование | ***JUnit, NUnit, Selenium, JaCoCo, TestNG*** |
| ***CI/CD*** | ***Jenkins, GitHub Actions, GitLab CI*** |
| Мониторинг | ***Prometheus, Grafana, Datadog, New Relic*** |
| Управление задачами | ***Jira, Trello, YouTrack*** |

**Вывод: *Метрики*** — *это* *инструмент обратной связи*. Они не заменяют опыт и интуицию разработчика, но позволяют объективно видеть, что происходит с проектом:

Помогают предсказывать сбои;  
Выявляют ***«горячие точки»*** в коде;  
Делают команды и процессы прозрачными;  
Улучшают качество продукта и удовлетворённость клиентов.

***Главное*** *—* ***использовать метрики с умом***: не ради галочки, а ради **улучшения качества разработки и продукта в целом.**