Otwarte repozytoria kodu i pomiar oprogramowania laboratorium

jaroslaw.hryszko@uj.edu.pl

Instytut Informatyki i Matematyki Komputerowej UJ

Semestr letni 2024/2025

Plan realizacji projektu „SkanUJkod”

1 Założenia

- Czas trwania: 12 tygodni
- Cel: stworzenie (w formie rozszerzalnego skanera) narzędzia do liczenia metryk kodu, metryk projektowych oraz pokrycia testowego.

2 Opis głównych funkcjonalności

- **Metryki statyczne kodu** – w oparciu o AST (drzewa składniowe) lub inne narzędzia parsujące.
- **Metryki projektowe (dane z Gita)** – m.in. liczba zmian pliku, liczba deweloperów, integracja z *just-in-time defect prediction*.
- **Wykrywanie code smells i potencjalnych luk bezpieczeństwa**.
- **Pokrycie testowe** – liczenie z wykorzystaniem instrumentacji kodu (np. statement coverage, branch coverage, condition coverage, MC/DC, prime path coverage, itp.).
- **Architektura łatwa do rozszerzenia na kolejne języki** – zgodnie z zasadą *Open-Closed Principle*.

---

3 Zadania i odpowiedzialności

Każde zadanie posiada lidera (odpowiedzialnego za koordynację), jednakże wszyscy członkowie zespołu współpracują przy realizacji poszczególnych zadań w miarę potrzeb.

1. **Zadanie 1: Opracowanie architektury skanera (tydzień 1–2)**
   Lider: Student 1
   Zakres:
   - Analiza wymagań i ustalenie głównych modułów systemu.
   - Zaprojektowanie architektury umożliwiającej dodawanie obsługi wielu języków i nowych metryk (architektura wtyczkowa).
   - Przygotowanie repozytorium, narzędzi CI/CD i standardów kodowania.
2. **Zadanie 2: Implementacja parsera/AST (tydzień 2–4)**
   Lider: Student 2
   Zakres:
   - Research dostępnych parserów (np. ANTLR) i wybór rozwiązania.
   - Implementacja mechanizmu parsującego kod w wybranym języku (np. Java).
   - Stworzenie interfejsów do analizy drzewa składniowego na potrzeby liczenia metryk.
3. **Zadanie 3: Metryki statyczne kodu (tydzień 3–5)**
   Lider: Student 3
   Zakres:
   - Implementacja obliczania podstawowych metryk statycznych (np. liczba linii kodu, złożoność cyklomatyczna, liczenie klas, metod, zależności).
   - Rozszerzenie modułu o bardziej zaawansowane metryki (np. kopiowanie kodu, zagnieżdżenia, itp.).
   - Konfiguracja wyjścia w formie raportu (CLI).
4. **Zadanie 4: Metryki projektowe na bazie Gita (tydzień 4–6)**
   Lider: Student 4
   Zakres:
   - Implementacja modułu do analizy historii zmian (logi gita).
   - Wyliczanie metryk: liczba deweloperów na plik, liczba commitów, częstotliwość zmian, itp.
   - Możliwe podpięcie się do metryk *just-in-time defect prediction* (np. metryki z pracy Kamei).
5. **Zadanie 5: Code smells i luki bezpieczeństwa (tydzień 5–7)**
   Lider: Student 5
   Zakres:
   - Opracowanie listy wybranych *smells* i podstawowych luk bezpieczeństwa.
   - Implementacja rozpoznawania wskazanych wzorców w kodzie (np. *long method*, *god class*, SQL injection).
   - Raportowanie poziomu zagrożenia / typów *smells*.

6. **Zadanie 6: Instrumentacja kodu i obliczanie pokrycia testowego (tydzień 6–8)**
   Lider: Student 6
   Zakres:

   - Research metod instrumentacji (np. bajtkod, wstrzykiwanie *probe points*).
   - Implementacja liczenia *statement coverage*, *branch coverage*, *condition coverage*, *MC/DC*, *prime path coverage* itp.
   - Generowanie raportów pokrycia i integracja z poprzednimi modułami.

7. **Zadanie 7: Integracja z CLI (tydzień 7–9)**
   Lider: Student 7
   Zakres:

   - Przygotowanie narzędzia w trybie *command line*, łączącego wszystkie funkcjonalności.
   - Logika konfiguracyjna (np. wybór ścieżki do kodu, wybór metryk).
   - Testy automatyczne (unit/integracyjne) potwierdzające spójność wszystkich modułów.

8. **Zadanie 8: Dokumentacja definicji metryk (tydzień 8–10)**
   Lider: Student 8
   Zakres:

   - Przygotowanie oficjalnego dokumentu opisującego definicje operacyjne metryk (jak liczymy każdą metrykę).
   - Tworzenie przykładów użycia, wskazówek interpretacji wyników.
   - Integracja dokumentacji technicznej (architektura, instrukcja kompilacji, uruchamianie).

9. **Zadanie 9: Opcjonalny GUI (tydzień 9–11)**
   Lider: Student 9
   Zakres:

   - Zaprojektowanie uproszczonego interfejsu graficznego (lub *webowego*) prezentującego wyniki skanowania.
   - Połączenie go z istniejącym CLI (np. wywoływanie komend w tle).
   - Testy użyteczności i wyglądu.

10. **Zadanie 10: Integracja końcowa, testy i dokumentacja (tydzień 11–12)**
    Lider: Student 10
    Zakres:

    - Ostateczne połączenie wszystkich modułów w jeden pakiet instalacyjny.
    - Wykonanie testów akceptacyjnych i końcowa weryfikacja jakości kodu.
    - Finalizacja kompletnej dokumentacji, z uwzględnieniem planów dalszego rozwoju.

4 Harmonogram realizacji

Tydzień	Zadania	Odpowiedzialni (propozycja)
1	▪ Rozpoczęcie Zadania 1 (architektura) ▪ Ustalenie repozytorium, CI/CD	Student 1 (lider) + cały zespół
2	▪ Finalizacja Zadania 1 ▪ Start Zadania 2 (parser/AST)	▪ Student 1 ▪ Student 2 (lider Zadania 2)
3	▪ Kontynuacja Zadania 2 ▪ Rozpoczęcie Zadania 3 (metryki statyczne)	▪ Student 2 ▪ Student 3 (lider Zadania 3)
4	▪ Finalizacja Zadania 2 ▪ Rozwój Zadania 3 (rozszerzanie metryk) ▪ Start Zadania 4 (metryki projektowe z Gita)	▪ Student 2 ▪ Student 3 ▪ Student 4 (lider Zadania 4)
5	▪ Kontynuacja Zadania 3 ▪ Rozwój Zadania 4 ▪ Rozpoczęcie Zadania 5 (code smells, security)	▪ Student 3 ▪ Student 4 ▪ Student 5 (lider Zadania 5)

Tydzień	Zadania	Odpowiedzialni (propozycja)
6	▪ Finalizacja Zadania 3 i 4 ▪ Kontynuacja Zadania 5 ▪ Start Zadania 6 (instrumentacja, coverage)	▪ Student 3 ▪ Student 4 ▪ Student 5 ▪ Student 6 (lider Zadania 6)
7	▪ Kontynuacja Zadania 5 i 6 ▪ Rozpoczęcie Zadania 7 (integracja CLI)	▪ Student 5 ▪ Student 6 ▪ Student 7 (lider Zadania 7)
8	▪ Finalizacja Zadania 5 ▪ Dalsza praca nad Zadaniem 6 i 7 ▪ Start Zadania 8 (dokumentacja metryk)	▪ Student 5 ▪ Student 6 ▪ Student 7 ▪ Student 8 (lider Zadania 8)
9	▪ Kontynuacja Zadania 6, 7, 8 ▪ Rozpoczęcie Zadania 9 (GUI, opcjonalne)	▪ Student 6 ▪ Student 7 ▪ Student 8 ▪ Student 9 (lider Zadania 9)

Tydzień	Zadania	Odpowiedzialni (propozycja)
10	▪ Finalizacja Zadania 6, 7, 8 ▪ Rozwój Zadania 9 ▪ Początek Zadania 10 (integracja końcowa)	▪ Student 6 ▪ Student 7 ▪ Student 8 ▪ Student 9 ▪ Student 10 (lider Zadania 10)
11	▪ Kontynuacja Zadania 9, intensywna integracja (Zadanie 10) ▪ Testy i poprawki błędów	▪ Student 9 ▪ Student 10 ▪ Wszyscy (wsparcie testów)
12	▪ Finalizacja Zadania 9 i 10 ▪ Testy akceptacyjne i przygotowanie dokumentacji końcowej	▪ Student 9 ▪ Student 10 ▪ Wszyscy (wsparcie testów)

5 Uwagi końcowe

- Plan może ulegać modyfikacjom w zależności od postępów i dostępności członków zespołu.
- Polecam cotygodniowe spotkania całego zespołu (poza piątkowymi zajęciami).
- Ważne jest prowadzenie dokumentacji technicznej i utrzymywanie *readme* w repozytorium.
- Należy uwzględnić zasady *Open-Closed Principle* w celu łatwej rozszerzalności o nowe języki i nowe metryki.