1 Opis technik używanych przez studentów z poprzedniego roku:

Studenci poprzedniego rocznika w analizie jakości kodu źródłowego wykorzystywali szeroki wachlarz narzedzi i technik. Dominowały narzedzia statycznej analizy kodu, takie jak SonarQube, Qodana, PMD, Checkstyle czy Pylint, które umożliwiały wykrywanie typowych błedów programistycznych, nieużywanych zmiennych oraz naruszeń standardów stylistycznych. SonarQube był szczególnie popularny ze wzgledu na szerokie spektrum wykrywanych błedów, w tym duplikacje kodu, luki bezpieczeństwa i tzw. code smells".

Do szczegółowych analiz metryk takich jak cyklomatyczna złożoność kodu, liczba parametrów funkcji oraz wykrywanie zduplikowanych fragmentów kodu używano narzedzi specjalistycznych jak Lizard, OCLint, cppdepend oraz FTA. Zauważono, że studenci czesto implementowali dodatkowe skrypty, np. z wykorzystaniem OCLint, by dostosować analizy do specyficznych wymagań projektowych.

W celu oceny pokrycia kodu testami jednostkowymi czesto stosowano biblioteke JaCoCo, dzieki której możliwe było precyzyjne określenie procentowego pokrycia kodu i wskazanie obszarów wymagajacych dodatkowych testów.

Podczas analizy wyników poszczególnych narzedzi studenci zauważyli znaczne rozbieżności w metrykach, wynikajace z różnych sposobów liczenia parametrów, linii kodu oraz traktowania komentarzy i bloków kodu. W efekcie czesto dochodziło do manualnej interpretacji wyników w celu ujednolicenia i uzyskania klarownej oceny jakości.

Techniki analizy statycznej okazały sie niezwykle skuteczne, jednak wymagały świadomego doboru narzedzi i ich konfiguracji, aby w pełni odpowiadały na potrzeby analizowanych projektów. Praktyka studentów wskazuje na konieczność zachowania ostrożności przy interpretacji wyników generowanych przez różne narzedzia, szczególnie przy porównaniach miedzyprojektowych lub miedzyjezykowych.