

## Program szkolenia:

# Przeglądy kodu, które realnie wpływają na proces tworzenia oprogramowania

### Informacje:

<b>Nazwa:</b>	<b>Przeglądy kodu, które realnie wpływają na proces tworzenia oprogramowania</b>
<b>Kod:</b>	<b>Craft-practices-review</b>
<b>Kategoria:</b>	Craftsmanship
<b>Grupa docelowa:</b>	
<b>Czas trwania:</b>	1 dzień
<b>Forma:</b>	30% wykłady / 70% warsztaty

Jak określić wspólny cel zespołu? Jak zdefiniować oczekiwania wobec kodu? Jak wykonać przegląd kodu, który będzie produktywny, motywujący do rozwoju i wniesie realne zmiany na poziomie jakości? Na te pytania odpowie doświadczony craftsman.

### Zalety szkolenia:

- Zorientowanie na konkretny cel
- Integracja zespołu
- Miękkie aspekty komunikacji

## Szczegółowy program:

### 1. Cel przeglądu kodu

#### 1.1. Cele nadrzędne

1.1.1. Wymiana wiedzy, znajdowanie bugów jako efekt uboczny

1.1.2. Znaleziony defekt nie powinien się już powtórzyć u danego programisty

#### 1.2. Założenia

1.2.1. Zespół a nie załoga

1.2.2. Chęć dzielenia się wiedzą

1.2.2.1. Dzielisz się

1.2.2.2. Kunsztem technicznym

1.2.3. Chęć rozwoju i doskonalenia

1.2.4. Każdy ma potencjał

1.2.5. Nie chodzi o konkurencję ale o grę do wspólnej bramki

#### 1.3. Problemy

1.3.1. Wspólne oczekiwania i ich mierzenie

1.3.2. Przekierowanie emocji z osób na problemy

### 2. Oczekiwania wobec kodu - jak je gromadzić i jednoznacznie wyrażać

#### 2.1. Code style - definicja jak być powinno

##### 2.1.1. Strategie

2.1.1.1. Przez podobieństwa

2.1.1.2. Przez różnice

##### 2.1.2. Przykłady

2.1.2.1. Obsługa błędów

2.1.2.2. Walidacja

2.1.2.3. Komentarze

2.1.2.4. Testy

2.1.2.5. Czytelność

2.1.3. Rozwijanie i ewolucja

2.1.3.1. Retrospektywy

2.2. Checklist - sterowanie procesem

2.2.1. Nawyk wykonywania przez autora przed przeglądem

2.2.2. Wyznaczenie stylu prowadzenia przeglądu

2.2.2.1. Kolejność i priorytety

2.2.2.2. Kilka przebiegów - każdy aspekt sprawdzany w izolacji

2.2.3. Reakcja na incydent

2.2.3.1. Przegląd listy i uaktualnienie

2.2.4. Przykładowe checklisty

2.2.5. Proces rozbudowy

### 3. Wykonanie przeglądu kodu - jak robić to skutecznie i efektywnie

3.1. Automatyzacja

3.1.1. Sonar jako część build - nie wykonuj pracy, którą mogą wykonać maszyny

3.1.1.1. Quality gate

3.1.1.2. Metryki - dlaczego, kiedy i po co mierzyć każdą z nich

3.2. Narzędzia

3.2.1. Crucible

3.2.2. Gerrit

3.2.3. Podejście zdalne: gitflow - pull request

3.3. Aspekty miękkie komunikacji

3.3.1. Feedback

3.3.2. Elementy Porozumienia bez przemocy

3.3.3. Idea egoless programming

3.4. Proces przeglądu

3.4.1. Zrozumieć kod bez pomocy autora

3.4.2. Przegląd z autorem

3.4.3. Dyskusja - jak nią kierować

3.4.4. Zbieranie defektów

3.5. Proces pielęgnacji przeglądu

3.5.1. Reagowanie na incydenty przez zmianę checklisty

3.5.2. Retrospektywy procesu przeglądu