BO·TT·EGA
IT minds

Program szkolenia:

Craftsmanship - przybornik profesjonalisty: najlepsze techniki developerskie i architektoniczne z wykorzystaniem ACE, Boost oraz Qt

Informacje:

Nazwa:	**Craftsmanship - przybornik profesjonalisty: najlepsze techniki developerskie i architektoniczne z wykorzystaniem ACE, Boost oraz Qt**
Kod:	**CCPP-craft-C++ Craft**
Kategoria:	Craftsmanship dla programistów C i C ++
Grupa docelowa:	developerzy
Czas trwania:	3 dni
Forma:	50% wykłady / 50% warsztaty

Szkolenie stanowi syntezę kluczowych elementów klasycznej i nowoczesnej inżynierii oprogramowania. Daje ogólny pogląd na praktyczne aspekty wykorzystania omawianych technik w projektach. Omawiane zagadnienia leżą u podstaw nowoczesnych frameworków i technologii (takich jak Qt, Boost czy ACE) – co zwiększa poziom ich zrozumienia i pozwala na świadome korzystanie.

Profesjonalista w naszym ujęciu:

- doskonale włada technikami developerskimi i stylami architektonicznymi,
- potrafi porozumieć się z biznesem przy pomocy technik DDD,
- dobiera właściwe narzędzie do klasy problemu
- dostarcza kod wysokiej jakości.

Szkolenie przeznaczone dla programistów i projektantów pragnących poszerzyć swe kompetencje w zakresie profesjonalnych technik zwiększających jakość kodu i projektu. Zdobyta wiedza przekłada się w praktyczny sposób na produktywność mierzoną w szerszej perspektywie czasu.

Zalety szkolenia:

- Sprawdzone techniki SOLID, GRASP i DDD
- Osadzenie technik w architekturze aplikacji i systemu
- Całość w kontekście testowania automatycznego
- Realne przykłady z wykorzystaniem Qt, Boost czy ACE

BO·TT·EGA
IT minds

Szczegółowy program:

1. Techniki Object Oriented

- 1.1. Ukierunkowanie myślenia w stylu OO
- 1.2. Code smell
- 1.3. Pułapki dziedziczenia
  - 1.3.1. Zamknięcie kodu na rozbudowę
  - 1.3.2. Zastępowania dziedziczenia kompozycją – praktyczne zalety zmiany podejścia
    - 1.3.2.1. Dziedziczenie nie nadaje się do modelowania ról
    - 1.3.2.2. Liskov Substitution Principle
    - 1.3.2.3. Wzorzec Party
    - 1.3.2.4. Wzorzec Role Object i Extension Object
- 1.4. Efektywne wykorzystanie Object Oriented
  - 1.4.1. GRASP
  - 1.4.2. SOLID
  - 1.4.3. RDD
- 1.5. GRASP - General Responsibility Assignment Software Patterns
- 1.6. Praktyczne wykorzystanie SOLID
  - 1.6.1. Kohezja klas
    - 1.6.1.1. Jak ją osiągnąć
    - 1.6.1.2. Jak wykrywać zapachy kodu
  - 1.6.2. Otwartość na rozbudowę
    - 1.6.2.1. Wzorzec Strategii
    - 1.6.2.2. 3 rodzaje logiki: stabilna, domknięcia, wybór domknięć
  - 1.6.3. Kiedy dziedziczenie nie ma sensu

BO·TT·EGA IT minds

4.1.2. Repository

4.1.3. ORM - zakres stosowalności

4.1.3.1. Pułapki Lazy Loading - wykrywanie i naprawa

4.1.3.2. Blokowanie optymistyczne

4.1.3.3. Uni of Work

4.2. Porządkowania logiki biznesowej (Servisy, DDD)

4.3. Inversion of Control – sprawdzona koncepcja budowy frameworków i systemów

4.3.1. Dependency Injection – podstawa współczesnych frameworków

4.3.1.1. Wsparcie dla testability

4.3.1.2. Praktyczne techniki wykorzystania w celu osiągnięcia giętkości designu

4.3.2. Systemy sterowane zdarzeniami

4.3.2.1. Architektura pluginowa

4.3.2.2. Separacja modułów

4.3.2.3. Zwiększanie responsywności systemu

4.3.2.4. Skalowanie

4.3.2.5. Wybrane przykłady z wykorzystaniem ACE, Boost oraz Qt

4.3.3. Wybrane architektury aplikacji

4.3.3.1. Klasyczna n-warstwowa architektura (odmiany)

4.3.3.2. Uproszczone architektury

4.3.3.3. Command-query Responsibility Segregation

4.3.3.4. Architektura wspierająca Domain Driven Design

4.3.3.5. Ultra-skalowalne systemy

5. Testability – projektowanie pod kątem wsparcia dla TDD

5.1. Podejście Specify First

BO·TT·EGA
IT minds