

Warsztat Zwinnego Programisty

Praktyczne Aspekty Rozwoju Oprogramowania

- Krzysztof Bulwiński i Wojciech Konopka
- 18-05-2016

Zwinny Programista

Acceptance Test
Driven Development

Test Driven
Development

Narzędzia
programistyczne

Zaawansowana
kontrola wersji



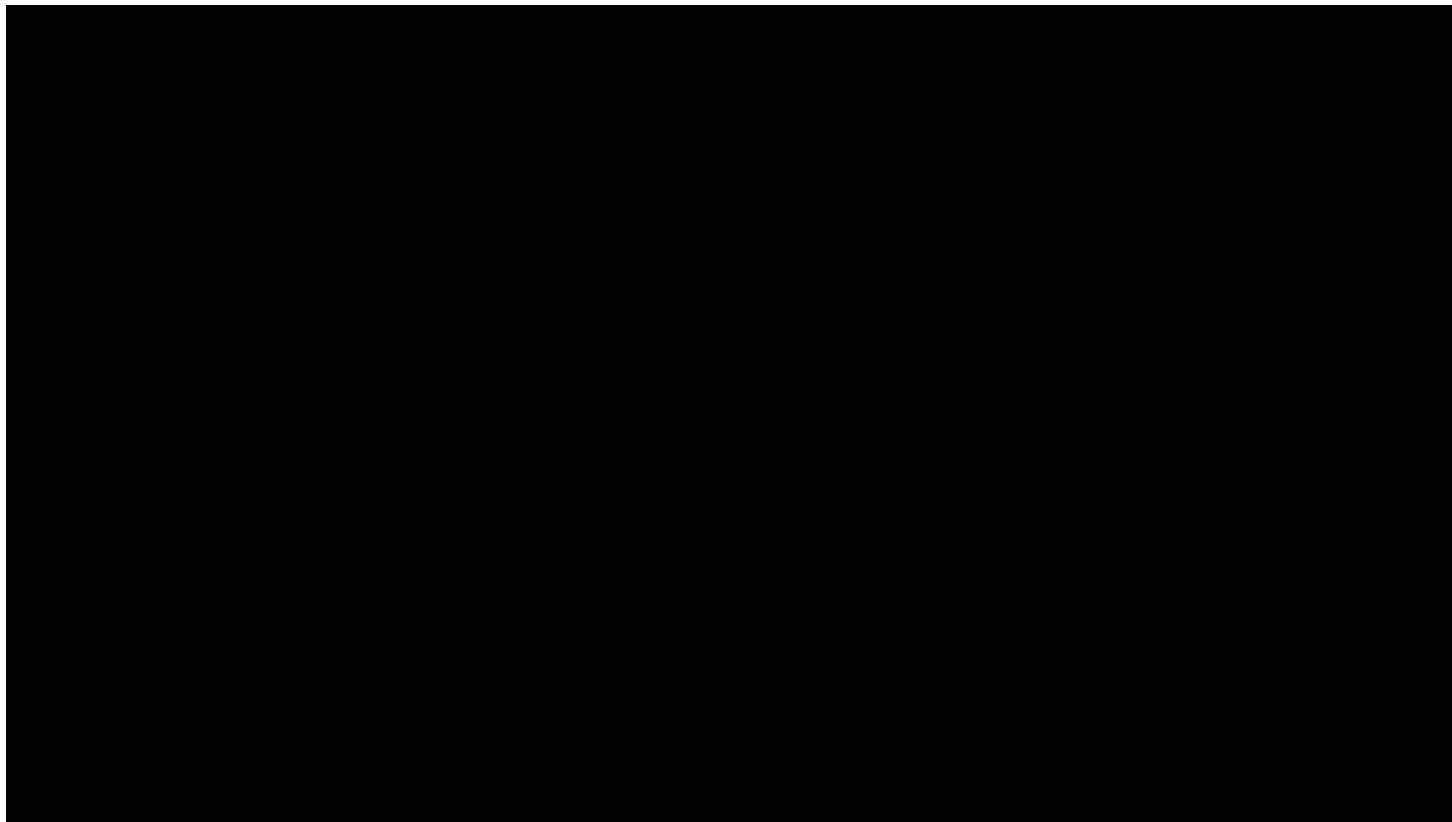
Ciągła Integracja

Programowanie w parach

Dostarczanie
oprogramowania

Code Review

Ciągła Integracja



Ciągła Integracja - CI

... ale po co to wszystko?

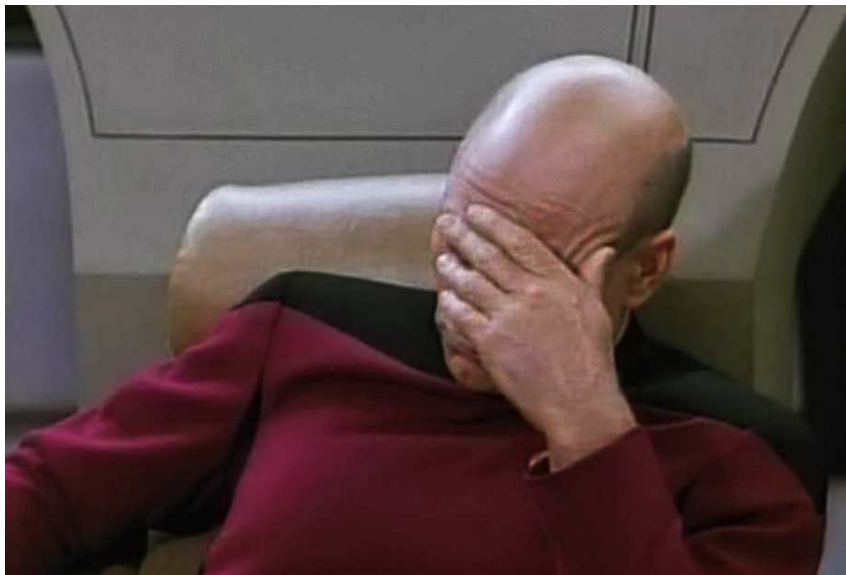


Świat bez Ciągłej Integracji

Programista



Klient



Ciągła Integracja

Czas ubrudzić sobie ręce 😊

DEMO

Benefity ciągłej integracji

- Szybsze dostarczanie oprogramowania
- Wiadomo, w którym momencie system przestaje działać
- Wychwytywanie błędów na wcześniejszych etapach procesu projektowego – poprawienie jest znacznie tańsze



Celem CI jest stała dostępność działającego oprogramowania

Wdrażanie Ciągłej Integracji

Czego potrzebujemy na samym początku?

1. Kontrola wersji
2. Automatyczna kompilacja
3. Zgoda zespołu



Wdrażanie Ciągłej Integracji

Kontrola wersji

Elementy projektu w repozytorium:

- Kod
- Testy
- Skrypty baz danych
- Skrypty kompilacji
- ...




Wdrażanie Ciągłej Integracji

Automatyczna kompilacja

Kompilacja z wiersza poleceń – dlaczego?

- Przeprowadzenie audytu
- Skrypty kompilacji traktowane jak baza kodu
- Ułatwia zrozumienie kompilacji, utrzymania i debugowania kompilacji



Wdrażanie Ciągłej Integracji

Zgoda Zespołu

Ciągła Integracja jest procedurą, nie narzędziem



Warunki wstępne ciągłej integracji

Ewidencjonuj regularnie

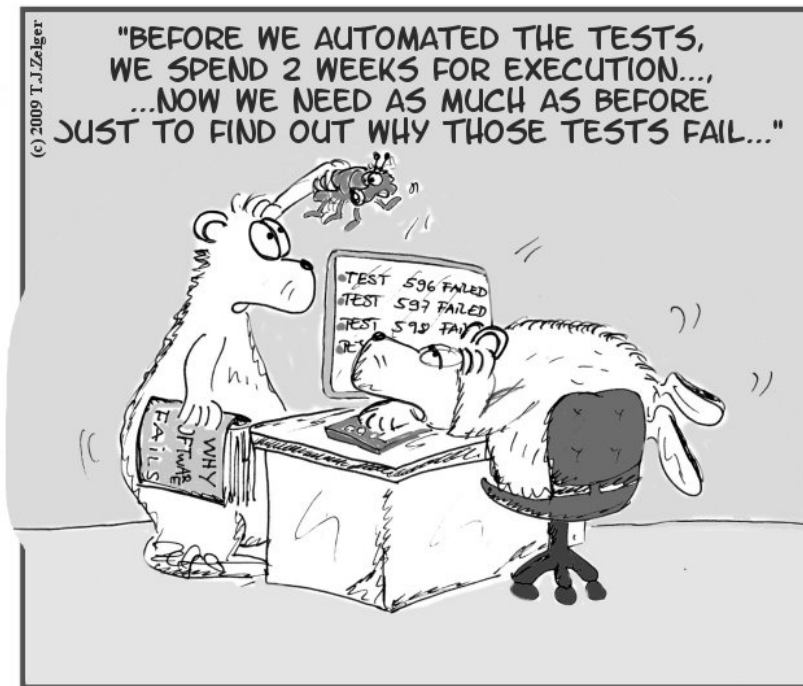
- Przynajmniej kilka razy dziennie
- Mniejsze prawdopodobieństwo zepsucia kompilacji
- Lepsze dyscyplina w kwestii refaktoryzacji
- W przypadku katastrofy (np. skasowania pliku) utrata niewielkiej ilości danych



Warunki wstępne ciągłej integracji

Kompleksowy zestaw TA

- Testy jednostkowe
- Testy integracji modułów
- Testy akceptacyjne



Test Automation side effect

Warunki wstępne ciągłej integracji

Kompleksowy zestaw TA – Testy jednostkowe

- Testy jednostkowe pisane są w celu przetestowania zachowania niewielkich fragmentów aplikacji (np. metody) w izolacji od pozostałych
- Zazwyczaj przeprowadza je się bez uruchamiania całej aplikacji
- Nie dotyczą bazy danych, systemu plików ani sieci
- Nie wymagają by aplikacja działała w środowisku zbliżonym do środowiska produkcyjnego
- Powinny przebiegać bardzo szybko

Warunki wstępne ciągłej integracji

Kompleksowy zestaw TA – Testy integracji modułów

- Sprawdzają zachowanie kilku modułów aplikacji
- Nie zawsze wymagają uruchomienia całej aplikacji
- Mogą dotyczyć bazy danych, systemu plików lub innych systemów
- Testy integracji modułów trwają zazwyczaj dłużej

Warunki wstępne ciągłej integracji

Kompleksowy zestaw TA – Testy akceptacyjne

- Sprawdzają czy aplikacja spełnia kryteria akceptacyjne ustalone przez biznes, zapewniając zarówno odpowiednią funkcjonalność, jak i cechy takie jak wydajność, dostępność, bezpieczeństwo, itd.
- Najlepiej jeżeli sprawdzają całą aplikację w środowisku produkcyjnym
- Ich przeprowadzenie może zabierać sporo czasu

Warunki wstępne ciągłej integracji

Proces kompilacji i testowania możliwie krótki



Jeżeli cały proces trwa za długo to:

- Ludzie przestaną wykonywać pełną kompilację i przeprowadzać testy – będzie coraz więcej nieudanych kompilacji
- Proces CI będzie zabierał tyle czasu, że pojawi się szereg zatwierdzonych zmian – nie wiadomo co popsuło kompilację
- Programiści będą ewidencjonowali rzadziej ze świadomością, że kompilacja i przeprowadzenie testów równa się w nieskończoność czekaniu

Warunki wstępne ciągłej integracji

Zarządzanie środowiskiem programistycznym

- Programiści powinni zawsze zaczynać kolejny etap pracy od znanej dobrej wersji
- Powinni być w stanie uruchomić kompilację, wykonać zautomatyzowane testy i wdrożyć aplikację w kontrolowanym przez nich środowisku - najlepiej gdyby mogli wykonać to na własnej lokalnej maszynie – współdzielone tylko w usprawiedliwionych przypadkach

Aby to osiągnąć:

1. Staranne zarządzanie konfiguracją – kod źródłowy, skrypty kompilacji, wdrożenia, itd..
2. Zarządzanie konfiguracją zależności stron trzecich (np. Maven)
3. Zautomatyzowane testy możliwe do odpalenia na maszynie programistycznej

Kluczowe praktyki

- Nie ewidencjonuj niczego w popsutej kompilacji
- Zawsze testuj lokalnie wszystkie zmiany przed ich zatwierdzeniem
- Nigdy nie idź do domu, dopóki kompilacja nie działa poprawnie
- Zawsze bądź przygotowany na powrót poprzednich wersji
- Ustaw sobie limit czasu na poprawki przed cofnięciem zmian
- Nie wyłączaj testów failujących
- Weź odpowiedzialność za szkody powstałe w wyniku zmian
- TDD

Poprzednicy CI



Zapraszamy do klawiatury



NOKIA