TEST-DRIVEN DEVELOPMENT IN EMBEDDED-C

CEHUG
ŁÓDŹ 16.11.2022

Maciej Wołosewicz

Lead Software Engineer at GlobalLogic

Po co tu jesteście
i co Wam da ta prezentacja?

Narzędzie dzięki któremu:

- Będziecie mieli odwagę pierwsi powiedzieć cześć PMowi.
-
-

Narzędzie dzięki któremu:

- Będziecie mieli odwagę pierwsi powiedzieć cześć PMowi.
- Pisać czysty, modularny i wolny od błędów kod.
-

Narzędzie dzięki któremu:

- Będziecie mieli odwagę pierwsi powiedzieć cześć PMowi.
- Pisać czysty, modularny i wolny od błędów kod.
- Znajdziecie przyjemność pracy w cudzym kodzie.

Dwa podejścia w programowaniu

Debug later

Debug later

- Wszyscy piszą kod jak by jutra miało nie być.
-
-
-
-
-

Debug later

- Wszyscy piszą kod jak by jutra miało nie być.
- Błędy się nawarstwiają.
-
-
-
-

Debug later

- Wszyscy piszą kod jak by jutra miało nie być.
- Błędy się nawarstwiają.
- Niektórzy mogą implementować swoją funkcjonalność w oparciu o nasze błędy.
-
-
-

Debug later

- Wszyscy piszą kod jak by jutra miało nie być.
- Błędy się nawarstwiają.
- Niektórzy mogą implementować swoją funkcjonalność w oparciu o nasze błędy.
- Zbliża się deadline i zaczynają nadgodziny.
-
-

Debug later

- Wszyscy piszą kod jak by jutra miało nie być.
- Błędy się nawarstwiają.
- Niektórzy mogą implementować swoją funkcjonalność w oparciu o nasze błędy.
- Zbliża się deadline i zaczynają nadgodziny.
- Zaczyna się unikanie PMa.
-

Debug later

- Wszyscy piszą kod jak by jutra miało nie być.
- Błędy się nawarstwiają.
- Niektórzy mogą implementować swoją funkcjonalność w oparciu o nasze błędy.
- Zbliża się deadline i zaczynają nadgodziny.
- Zaczyna się unikanie PMa.
- Podczas standupów zaczyna nam zrywać neta.

Brzmi znajomo?

Dlaczego tak się dzieje?

- Pisanie kodu to trudny i złożony proces.
-
-
-
-

Dlaczego tak się dzieje?

- Pisanie kodu to trudny i złożony proces.
- Ludzie popełniają błędy.
-
-
-

Dlaczego tak się dzieje?

- Pisanie kodu to trudny i złożony proces.
- Ludzie popełniają błędy.
- Zleceniodawcy chcą wszystkiego coraz szybciej.
-
-

Dlaczego tak się dzieje?

- Pisanie kodu to trudny i złożony proces.
- Ludzie popełniają błędy.
- Zleceniodawcy chcą wszystkiego coraz szybciej.
- Presja czasu rośnie.
-

Dlaczego tak się dzieje?

- Pisanie kodu to trudny i złożony proces.
- Ludzie popełniają błędy.
- Zleceniodawcy chcą wszystkiego coraz szybciej.
- Presja czasu rośnie.
- Zaczyna się zmęczenie i irytacja.

Czy jest jakaś szansa,
żeby było lepiej?

Co nam daje TDD?

Co nam daje TDD?

- Natychmiastowe info o regresji.
-
-
-

Co nam daje TDD?

- Natychmiastowe info o regresji.
- Mniej bugów w modułach które tworzymy.
-
-

Co nam daje TDD?

- Natychmiastowe info o regresji.
- Mniej bugów w modułach które tworzymy.
- Lepszy design i modularyzacja.
-

Co nam daje TDD?

- Natychmiastowe info o regresji.
- Mniej bugów w modułach które tworzymy.
- Lepszy design i modularyzacja.
- Gotowy przykład użycia, co jest lepsze niż dokumentacja.

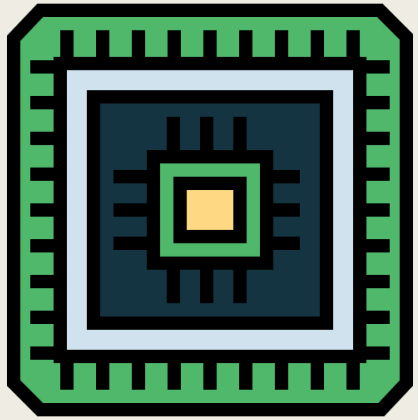
Co nam daje TDD?

- Natychmiastowe info o regresji.
- Mniej bugów w modułach które tworzymy.
- Lepszy design i modularyzacja.
- Gotowy przykład użycia, co jest lepsze niż dokumentacja.

I jeszcze więcej w embedded!

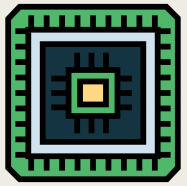
Dual-targeting

Dual-targeting



VS

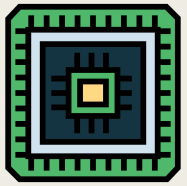
Dual-targeting



vs

- Możemy odpalić zarówno na PC jak i na Targecie
-
-
-
-
-

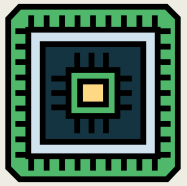
Dual-targeting



vs

- Możemy odpalić zarówno na PC jak i na Targecie
- Nie musimy za każdym razem programować procka.
-
-
-
-

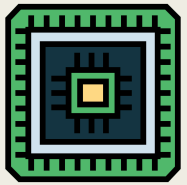
Dual-targeting



vs

- Możemy odpalić zarówno na PC jak i na Targecie
- Nie musimy za każdym razem programować procka.
- Wymusza architektoniczną separację SW od HW.
-
-
-

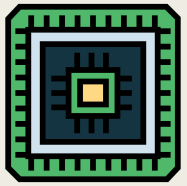
Dual-targeting



vs

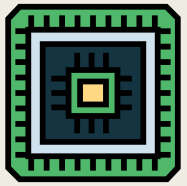
- Możemy odpalić zarówno na PC jak i na Targecie
- Nie musimy za każdym razem programować procka.
- Wymusza architektoniczną separację SW od HW.
- Dostajemy w gratisie wieloplatformowość!
-
-

Dual-targeting


vs


- Możemy odpalić zarówno na PC jak i na Targecie
- Nie musimy za każdym razem programować procka.
- Wymusza architektoniczną separację SW od HW.
- Dostajemy w gratisie wieloplatformowość.
- Nie musimy czekać na chłopaków z HW aż skończą (ani PMa aż nam zamówi evela)
-

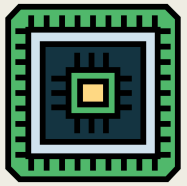
Dual-targeting


vs


- Możemy odpalić zarówno na PC jak i na Targecie
- Nie musimy za każdym razem programować procka.
- Wymusza architektoniczną separację SW od HW.
- Dostajemy w gratisie wieloplatformowość.
- Nie musimy czekać na chłopaków z HW aż skończą (ani PMa aż nam zamówi evela)
- Rozdzielamy bugi SW od bugów HW.

Mamy przez jakiś czas biurko puste jak Javowcy.

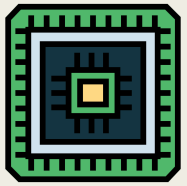
Dual-targeting



vs

Sen embeddedowca

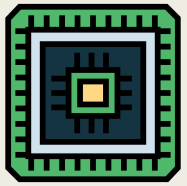
Dual-targeting



vs

- Budujemy image.
-
-
-

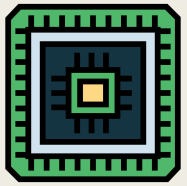
Dual-targeting



vs

- Budujemy image.
- Rozpakowujemy nową pachnącą płytkę ewaluacyjną.
-
-

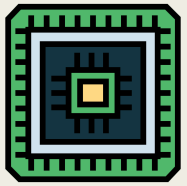
Dual-targeting



vs

- Budujemy image.
- Rozpakowujemy nową pachnącą płytkę ewaluacyjną.
- Ładujemy image na target.
-

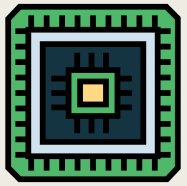
Dual-targeting



vs

- Budujemy image.
- Rozpakowujemy nową pachnącą płytkę ewaluacyjną.
- Ładujemy image na target.
- I.....

Dual-targeting

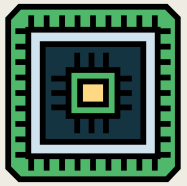


VS

- Budujemy image.
- Rozpakowujemy nową pachnącą płytkę ewaluacyjną.
- Ładujemy image na target.
- I.....

Działa za pierwszym razem.

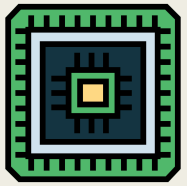
Dual-targeting



vs

Niestety są też ~~problemy~~ challenge.

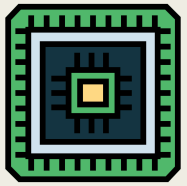
Dual-targeting - wyzwania



vs

- Kompilatory na PC i TARGET różnią się.
-
-
-
-

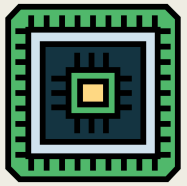
Dual-targeting - wyzwania



vs

- Kompilatory na PC i TARGET różnią się.
- Nie wszystkie featury są dostępne na TARGECIE.
-
-
-

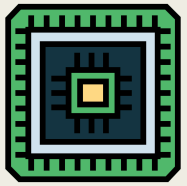
Dual-targeting - wyzwania



vs

- Kompilatory na PC i TARGET różnią się.
- Nie wszystkie featury są dostępne na TARGECIE.
- Może wystąpić różnica w endianesach.
-
-

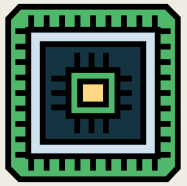
Dual-targeting - wyzwania



vs

- Kompilatory na PC i TARGET różnią się.
- Nie wszystkie featury są dostępne na TARGECIE.
- Może wystąpić różnica w endianesach.
- Może wystąpić różnica w wielkości typów podstawowych.
- I podobne...

Dual-targeting - wyzwania



vs

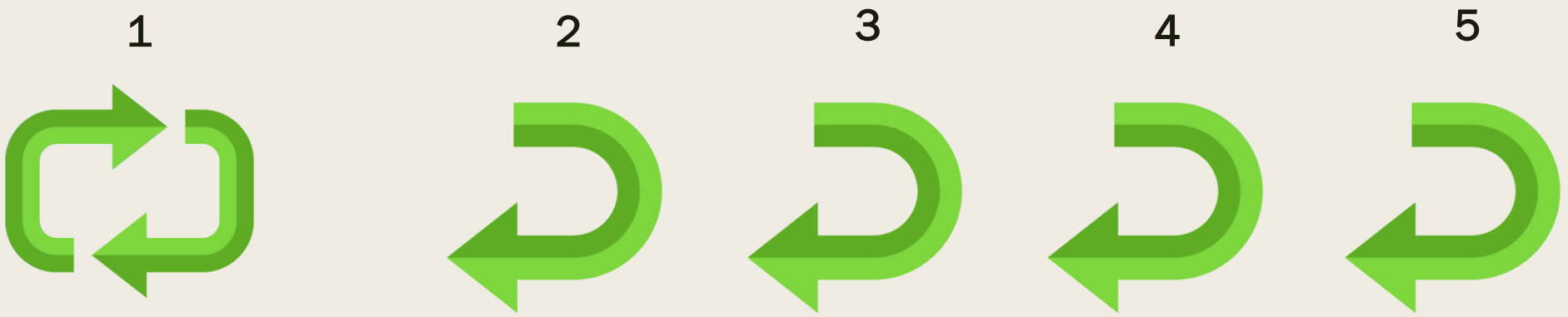
- Kompilatory na PC i TARGET różnią się.
- Nie wszystkie featury są dostępne na TARGECIE.
- Może wystąpić różnica w endianesach.
- Może wystąpić różnica w wielkości typów podstawowych.
- I podobne...

Jak to ogarnąć?

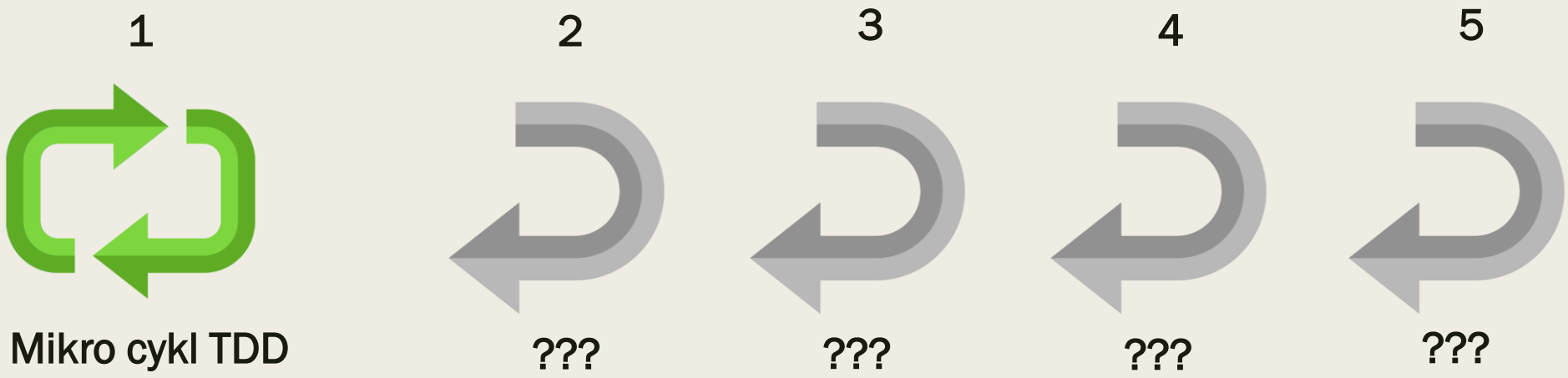
Dual-targeting

Cykl TDD w embedded

Cykl TDD w embedded

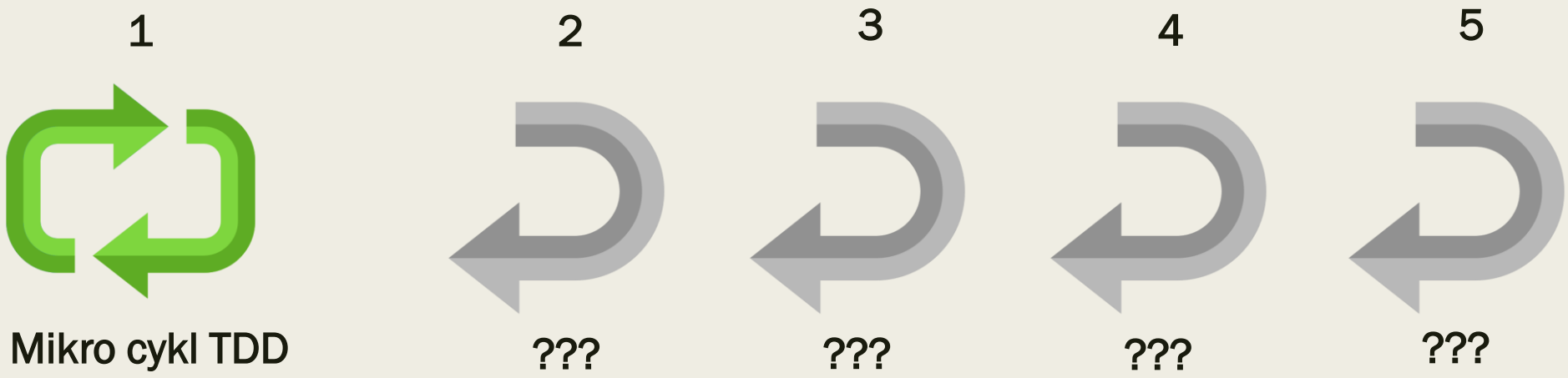


Cykl TDD w embedded



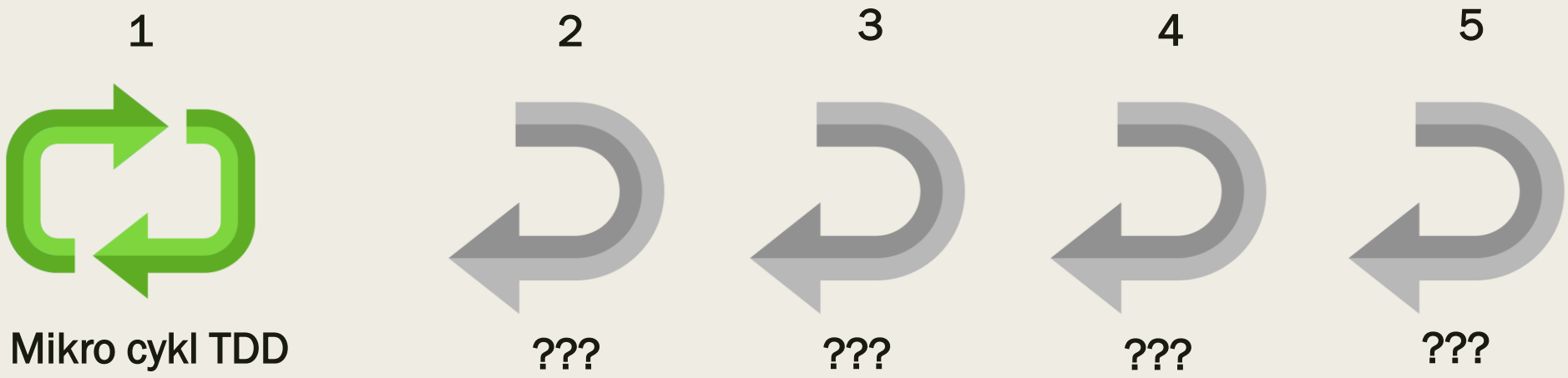
- Używamy non-stop.
-
-
-

Cykl TDD w embedded



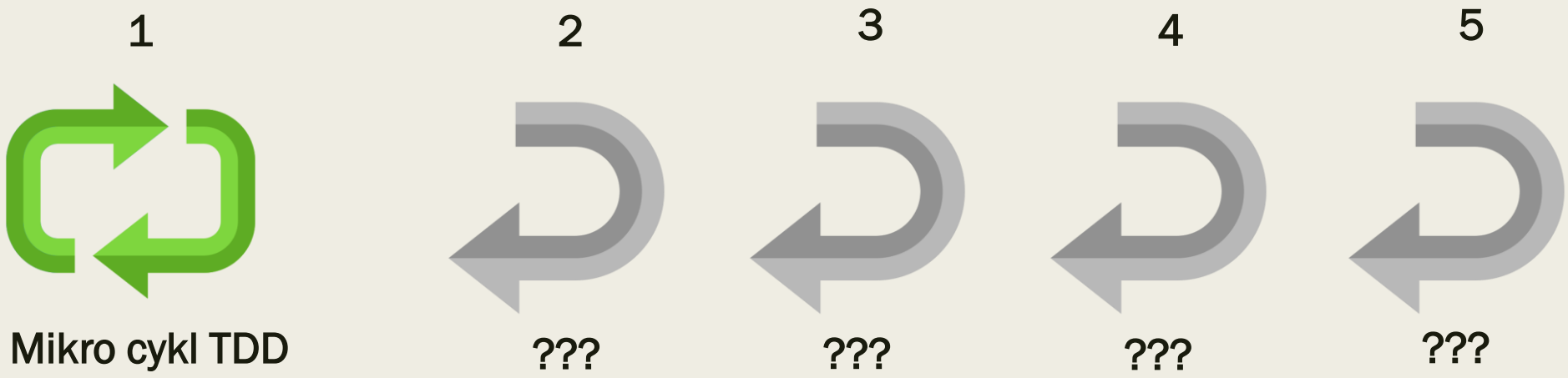
- Używamy non-stop.
- Valgrind, gcov, profile.
-
-

Cykl TDD w embedded



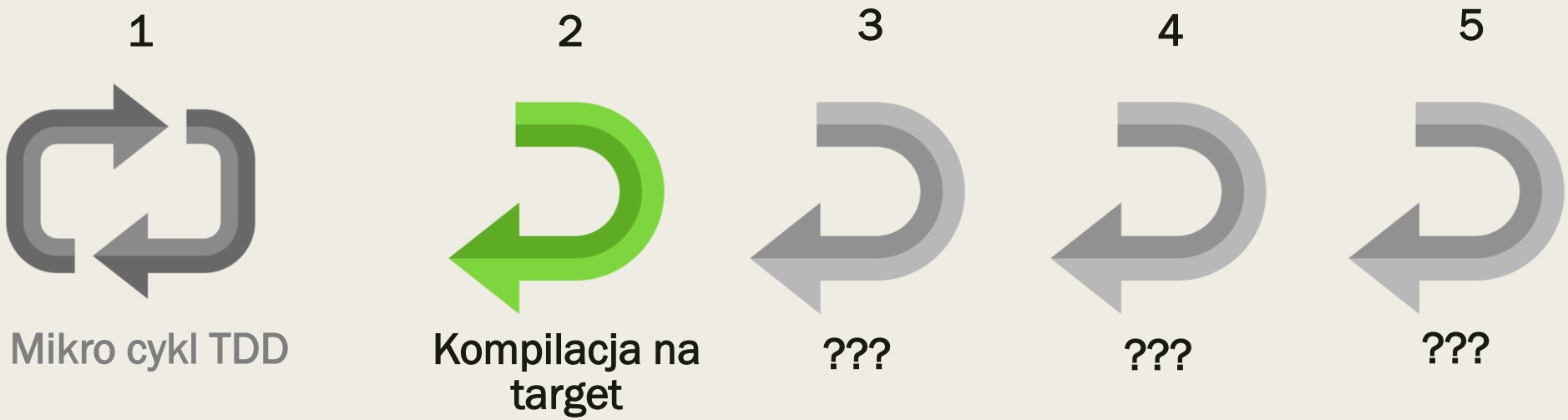
- Używamy non-stop.
- Valgrind, gcov, profile.
- Łatwiesze debugowanie.
-

Cykl TDD w embedded



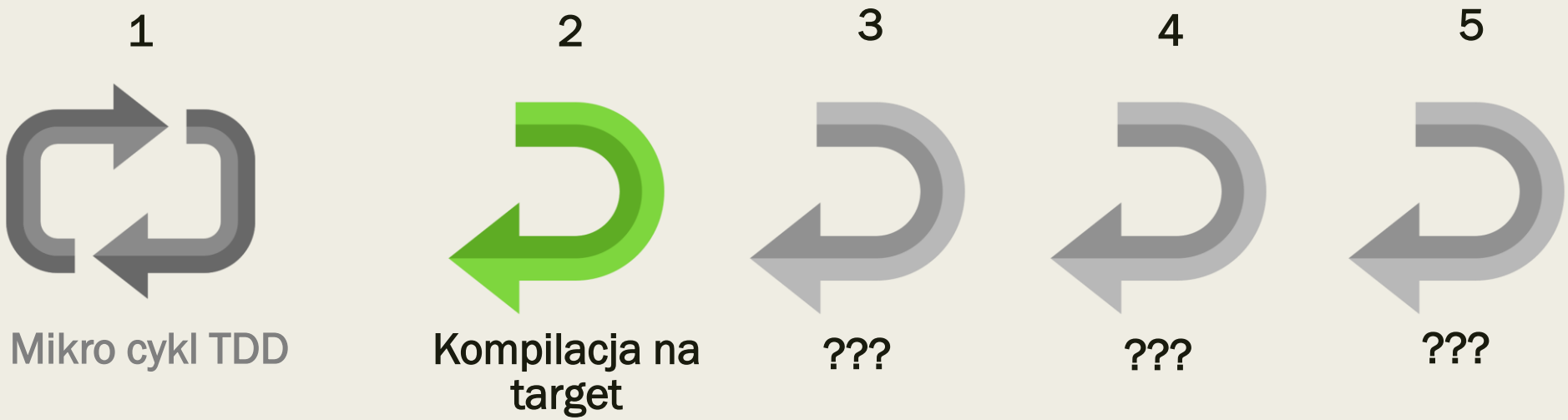
- Używamy non-stop.
- Valgrind, gcov, profile.
- Łatwiesze debugowanie.
- Szukamy okazji do modulacji i separacji od HW.

Cykl TDD w embedded



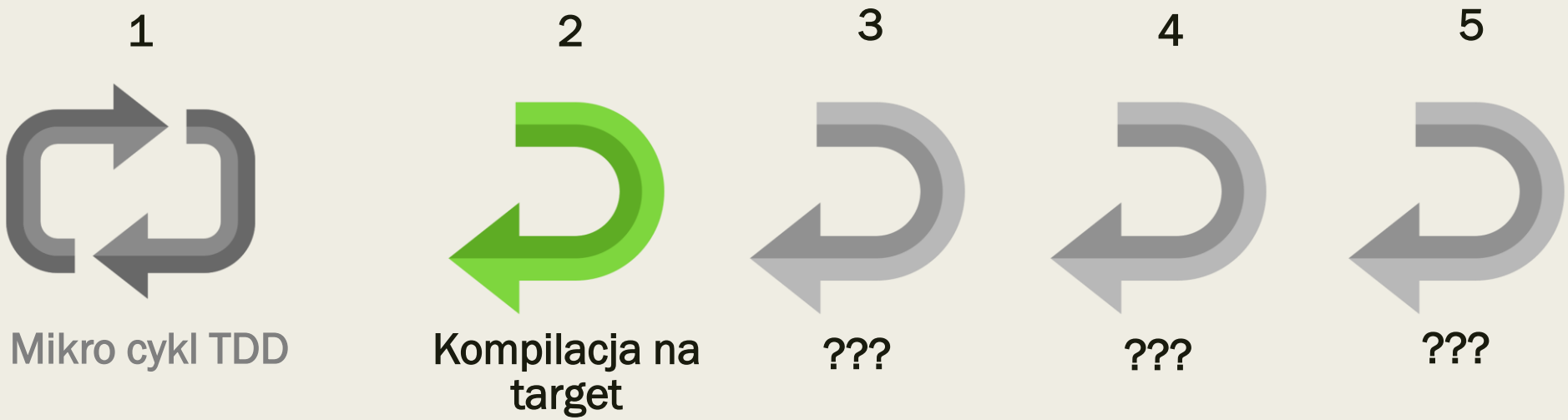
- Używamy co jakiś czas, albo CI co build.
-
-
-

Cykl TDD w embedded



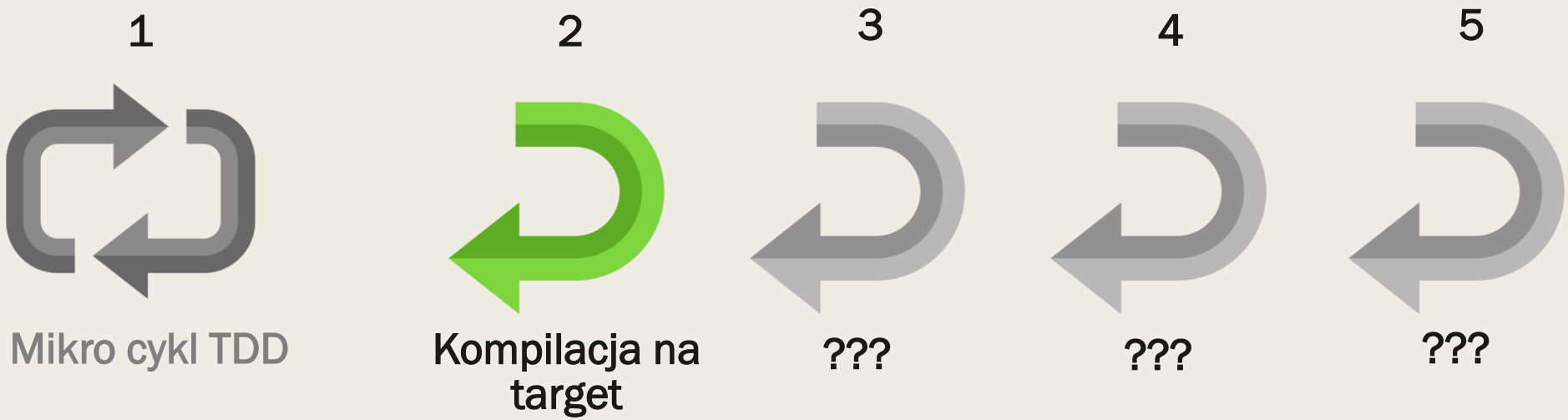
- Używamy co jakiś czas, albo CI co build.
- Używamy nowego includa.
-
-

Cykl TDD w embedded



- Używamy co jakiś czas, albo CI co build.
- Używamy nowego includa.
- Używamy nowego featura językowego.
-

Cykl TDD w embedded



- Używamy co jakiś czas, albo CI co build.
- Używamy nowego includa.
- Używamy nowego featura językowego.
- Wołamy zewnętrzną bibliotekę.

Cykl TDD w embedded

- Co jakiś czas.
-
-
-
-

Cykl TDD w embedded

- Co jakiś czas.
- Od razu widać błędy runtime.
-
-
-

Cykl TDD w embedded

- Co jakiś czas.
- Od razu widać błędy runtime.
- Widzimy różnicę w architekturze.
-
-

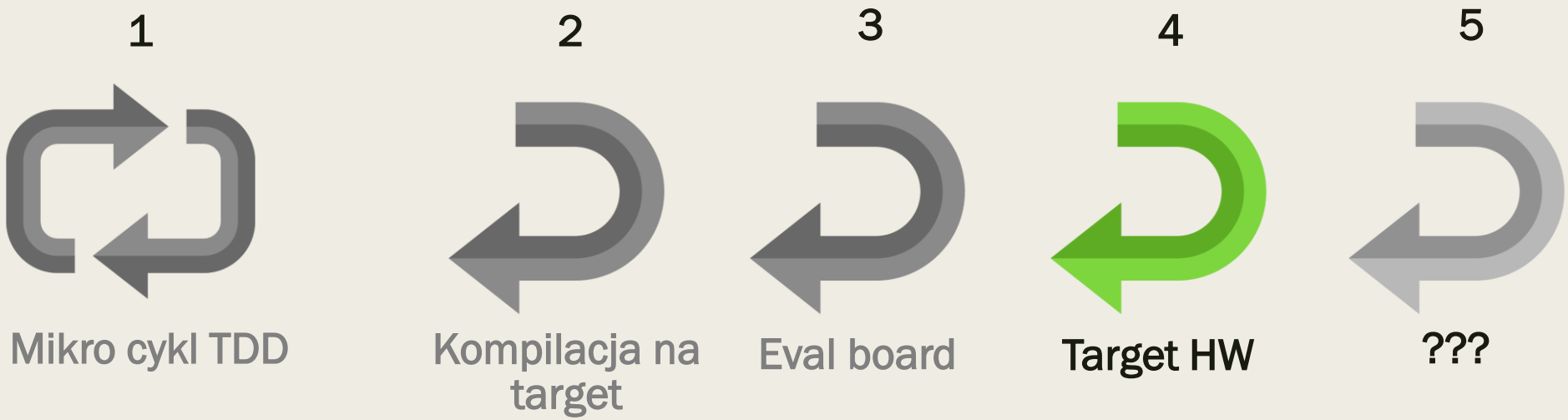
Cykl TDD w embedded

- Co jakiś czas.
- Od razu widać błędy runtime.
- Widzimy różnicę w architekturze.
- Dużo testpointów.
-

Cykl TDD w embedded

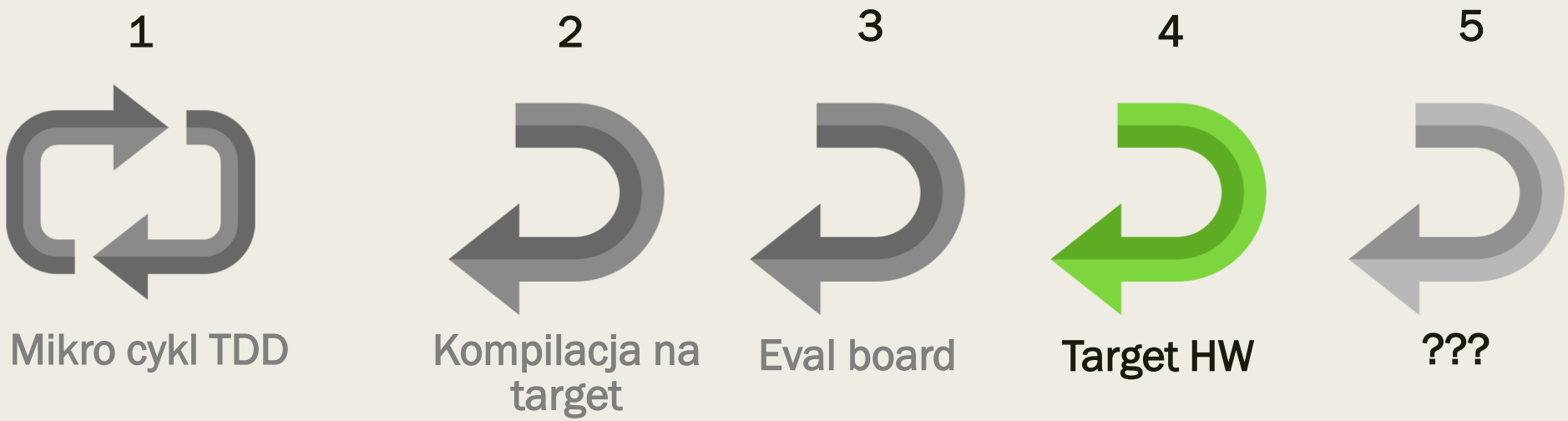
- Co jakiś czas.
- Od razu widać błędy runtime.
- Widzimy różnicę w architekturze.
- Dużo testpointów.
- Łatwa instumentacja.

Cykl TDD w embedded



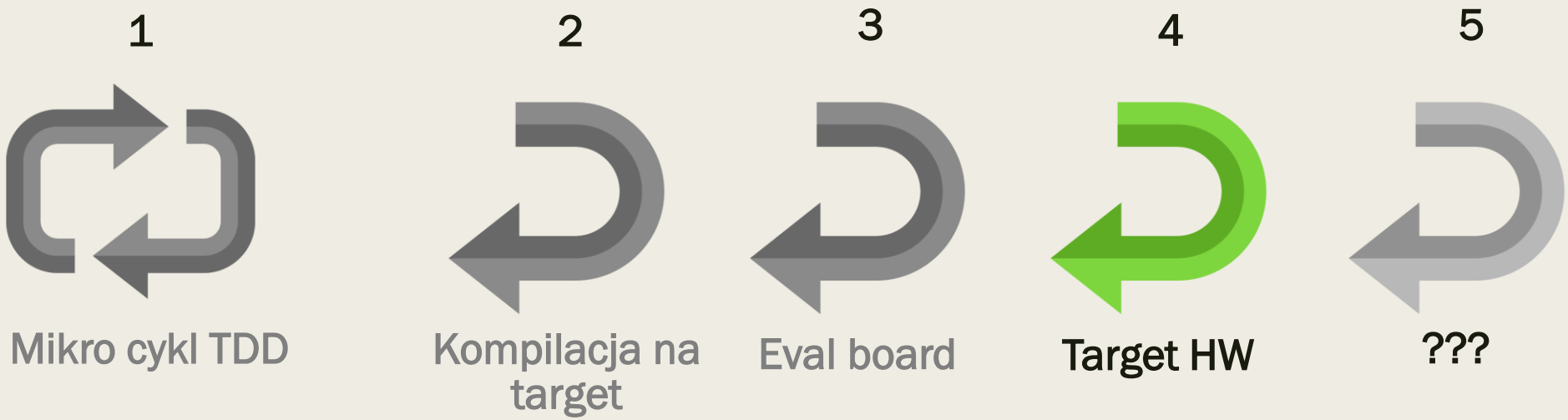
- Rzadziej niż na evalu.
-
-
-

Cykl TDD w embedded



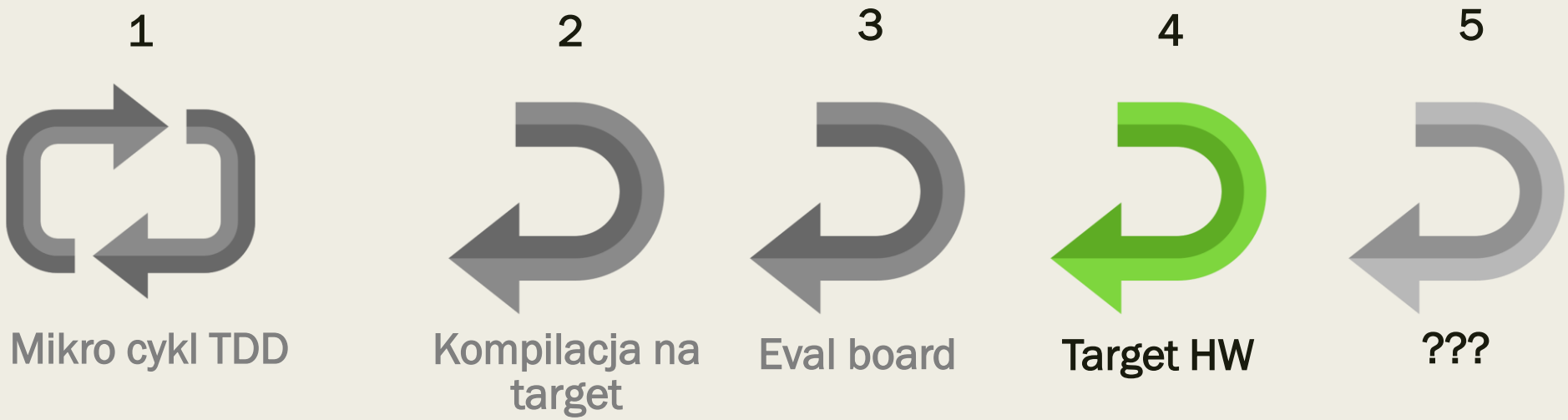
- Rzadziej niż na evalu.
- Możemy mieć do czynienia z jakimś bugiem HW.
-
-

Cykl TDD w embedded



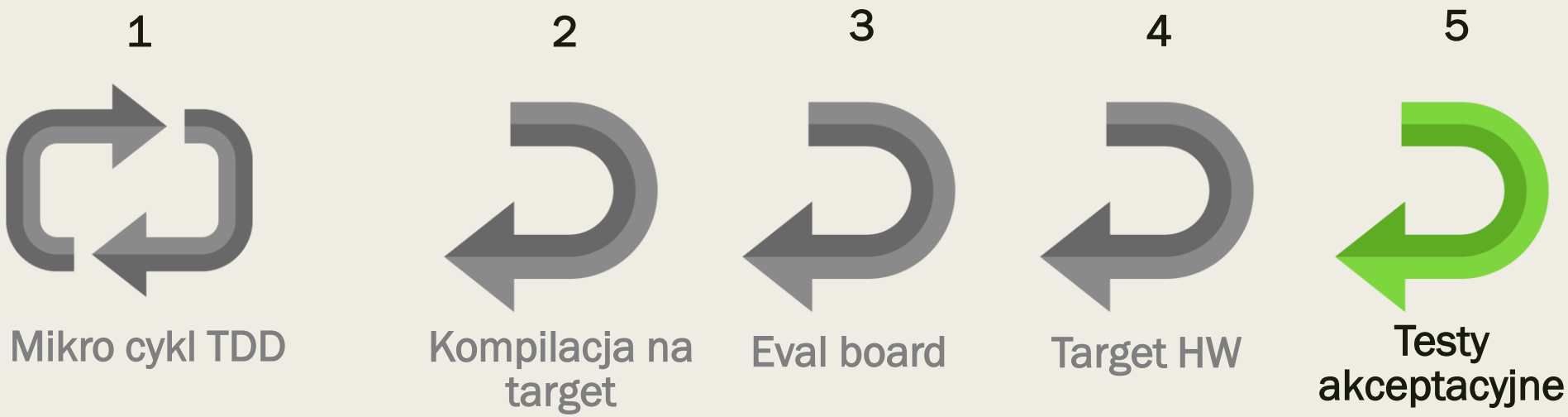
- Rzadziej niż na evalu.
- Możemy mieć do czynienia z jakimś bugiem HW.
- Możemy mieć do czynienia z jakimś specyficznym peryferium jak FPGA.
-

Cykl TDD w embedded



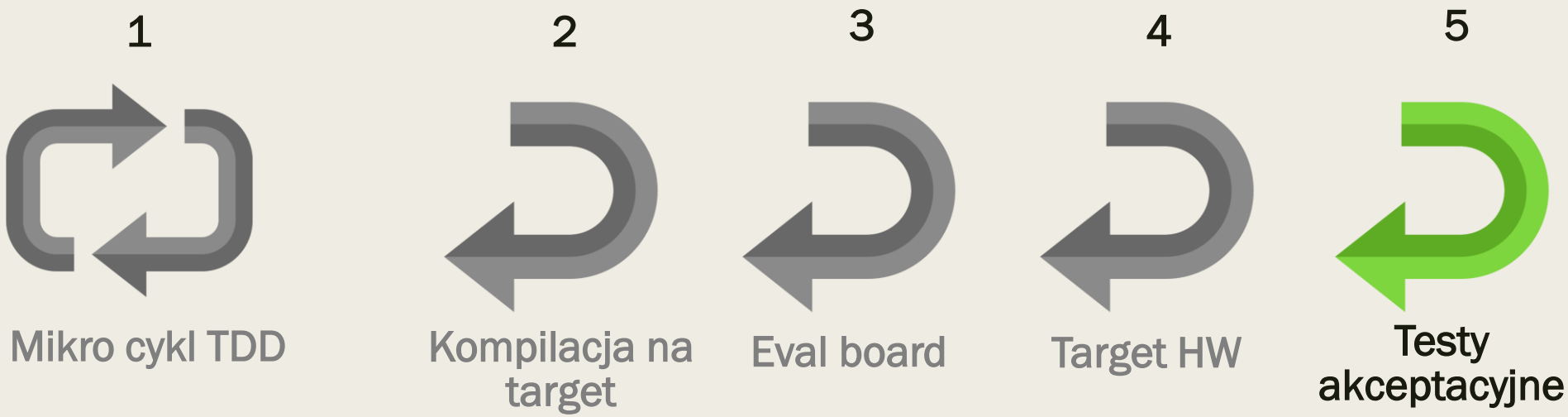
- Rzadziej niż na evalu.
- Możemy mieć do czynienia z jakimś bugiem HW.
- Możemy mieć do czynienia z jakimś specyficznym peryferium jak FPGA.
- Ograniczenia z pamięcią (embedded to nie tylko mikrokontrolery).

Cykl TDD w embedded



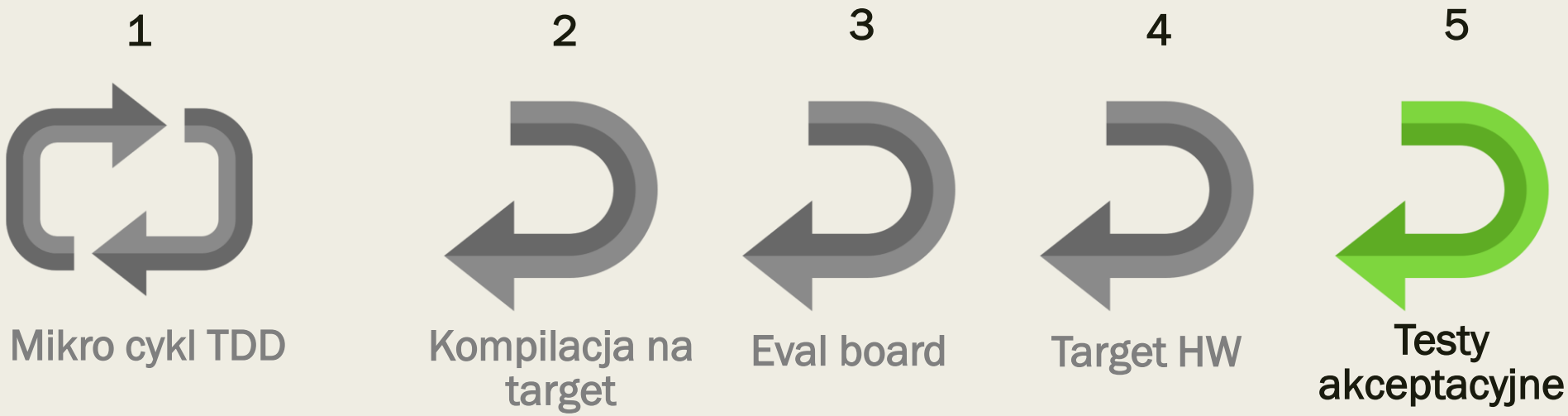
- Nie wszystko może być przetestowane automatycznie.
-
-

Cykl TDD w embedded



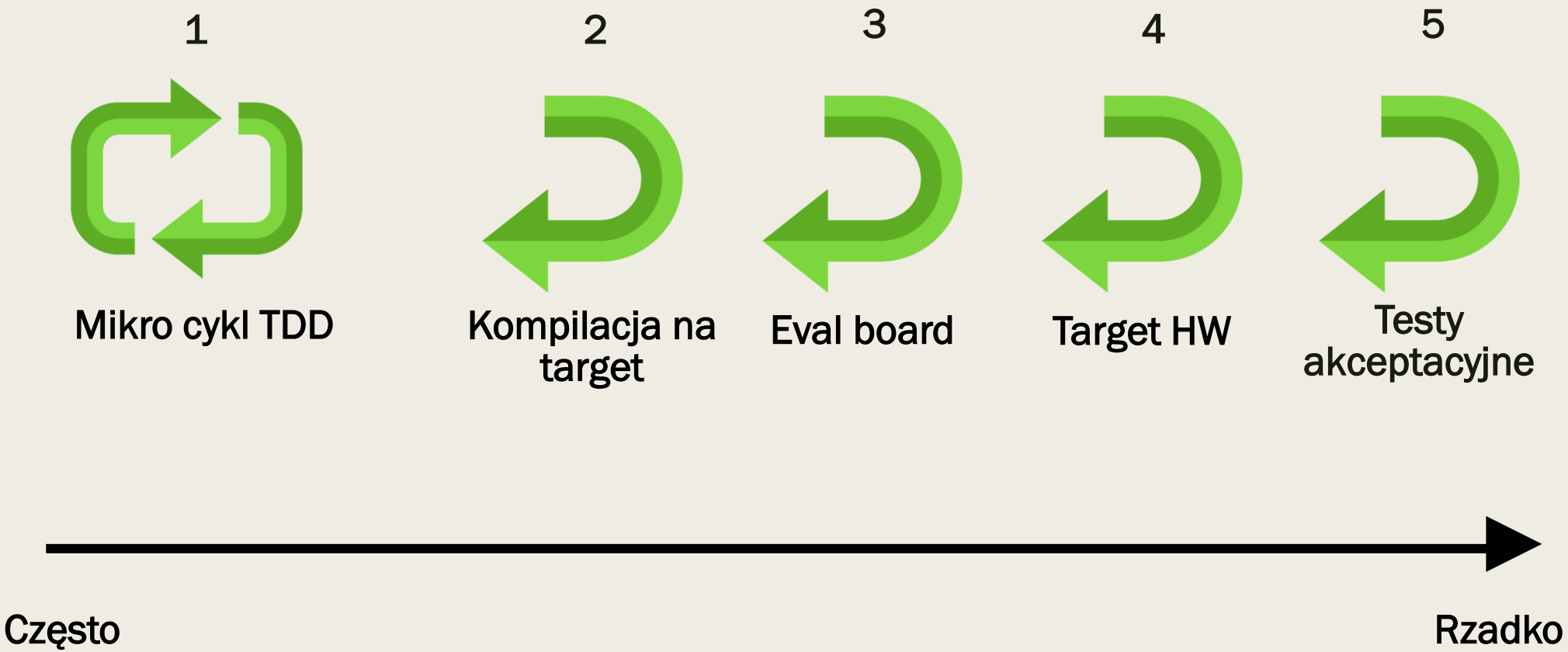
- Nie wszystko może być przetestowane automatycznie.
- Należy przekupić testerów manualnych.
-

Cykl TDD w embedded

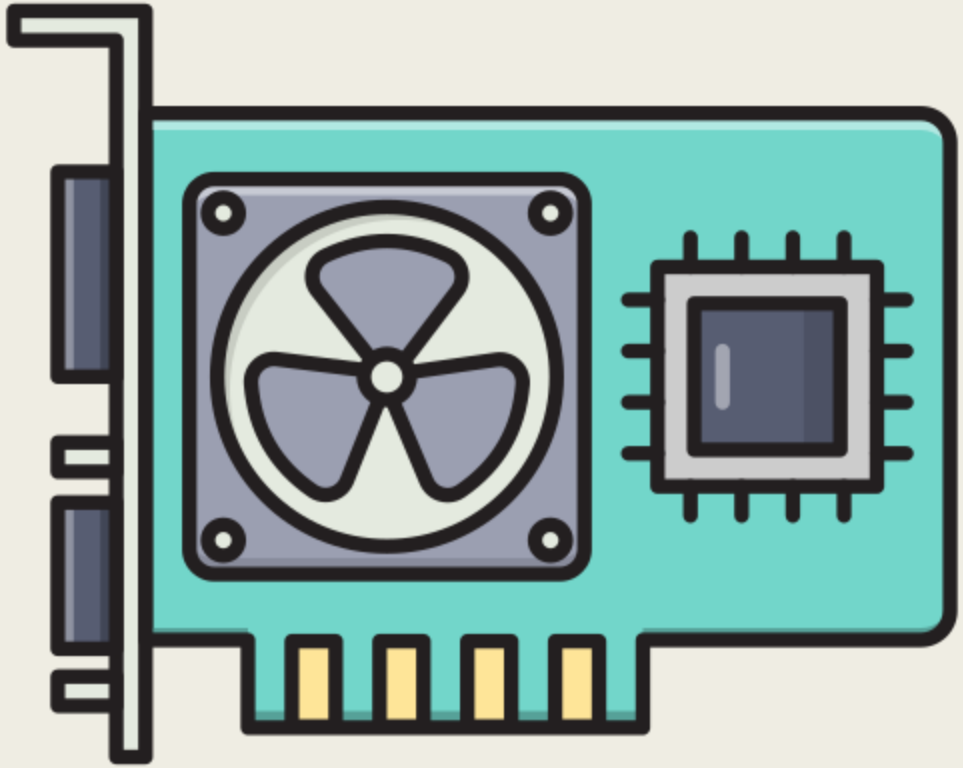


- Nie wszystko może być przetestowane automatycznie.
- Należy przekupić testerów manualnych.
- Robimy rzadko bo duży koszt testów manualnych.

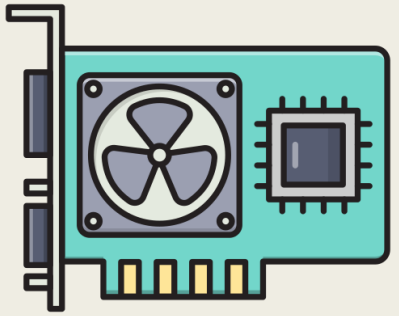
Cykl TDD w embedded



Testowanie hardwareu



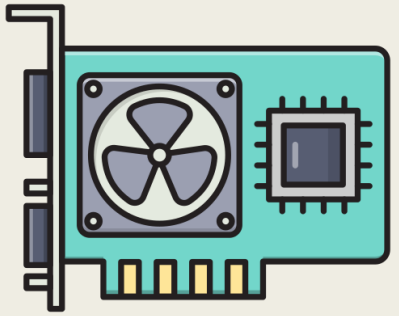
Testowanie hardwareu



??? ??? ???

3 rodzaje testów hardwaeru

Testowanie hardwareu

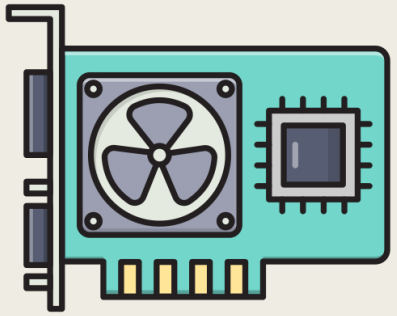


Testy automatyczne

???

???

Testowanie hardwareu



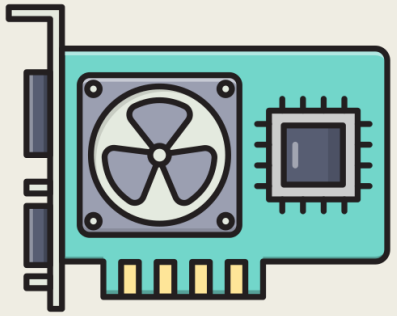
Testy automatyczne

???

???

- Pomagają zrozumieć działanie procesora.
-
-
-

Testowanie hardwareu



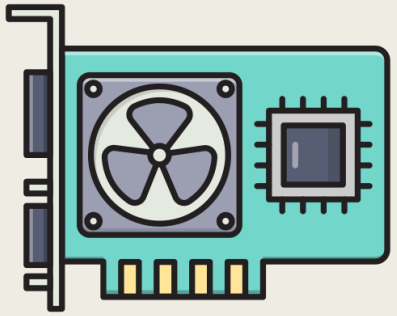
Testy automatyczne

???

???

- Pomagają zrozumieć działanie procesora.
- Pomagają zrozumieć działanie urządzeń peryferyjnych.
-
-

Testowanie hardwareu



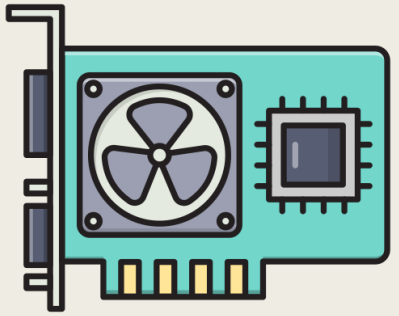
Testy automatyczne

???

???

- Pomagają zrozumieć działanie procesora.
- Pomagają zrozumieć działanie urządzeń peryferyjnych.
- **Od razu widać bugi między rewizjami.**
-

Testowanie hardwareu



Testy automatyczne

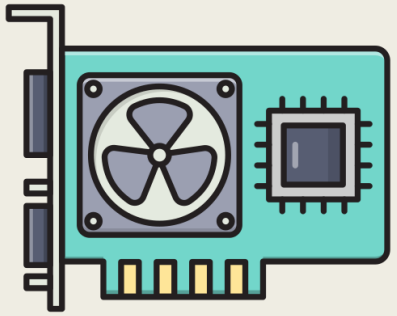
???

???

I można je wytknąć hardwaerowcom.

- Pomagają zrozumieć działanie procesora.
- Pomagają zrozumieć działanie urządzeń peryferyjnych.
- **Od razu widać bugi między rewizjami.**
-

Testowanie hardwareu



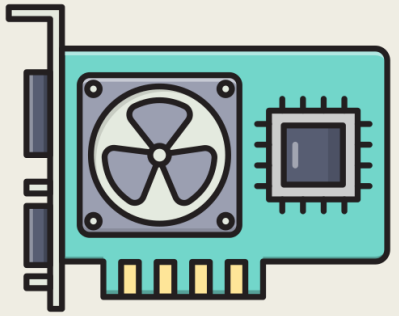
Testy automatyczne

???

???

- Pomagają zrozumieć działanie procesora.
- Pomagają zrozumieć działanie urządzeń peryferyjnych.
- Od razu widać bugi między rewizjami.
- Soft do certyfikacji EM w GRATIS!

Testowanie hardwareu

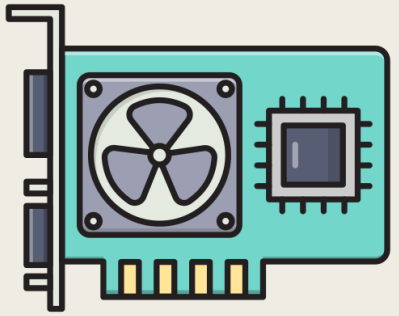


Testy automatyczne

Testy pół automatyczne

???

Testowanie hardwareu



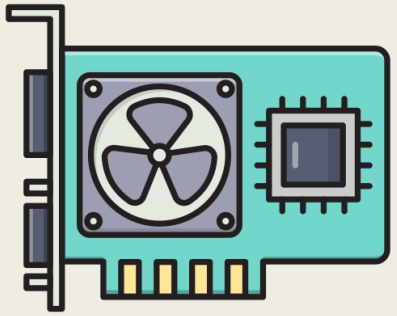
Testy automatyczne

Testy pół automatyczne

???

- Są drogie – używamy rzadko.
-
-

Testowanie hardwareu



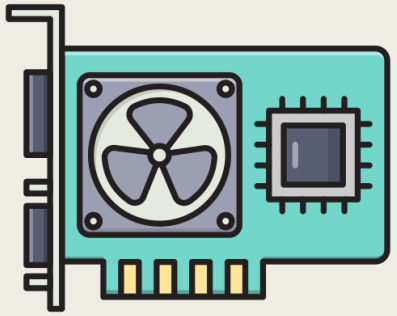
Testy automatyczne

Testy pół automatyczne

???

- Są drogie – używamy rzadko.
- Nie wszystko się da przetestować automatycznie (LEDy, wyświetlacze, itd.)
-

Testowanie hardwareu



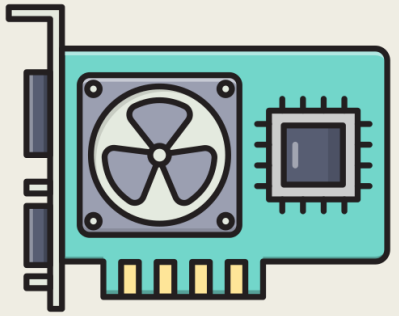
Testy automatyczne

Testy pół automatyczne

???

- Są drogie – używamy rzadko.
- Nie wszystko się da przetestować automatycznie (LEDy, wyświetlacze, itd.)
- Weryfikacja HW na produkcji.

Testowanie hardwareu

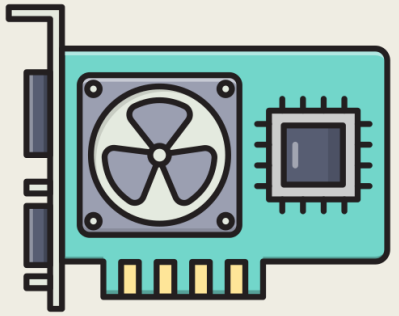


Testy automatyczne

Testy pół automatyczne

Automatyczne z zewnętrzną instrumentacją

Testowanie hardwareu



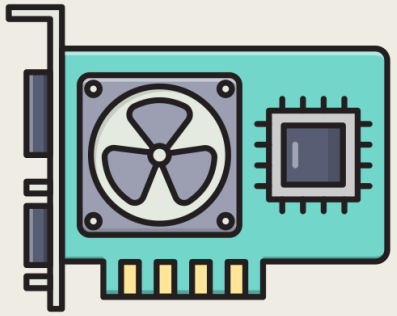
Testy automatyczne

Testy pół automatyczne

Automatyczne z zewnętrzną instrumentacją

- Możemy wymuszać wartości graniczne w sposób automatyczny.
-
-
-
-

Testowanie hardwareu



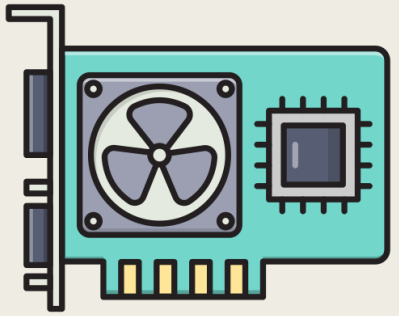
Testy automatyczne

Testy pół automatyczne

Automatyczne z zewnętrzną instrumentacją

- Możemy wymuszać wartości graniczne w sposób automatyczny.
- Łoża igłowe.
-
-
-

Testowanie hardwareu



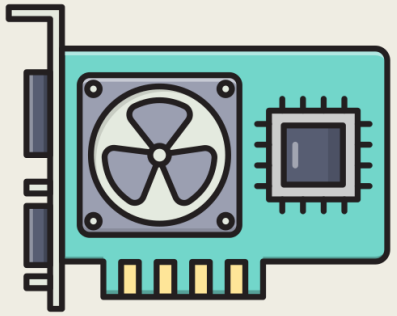
Testy automatyczne

Testy pół automatyczne

Automatyczne z zewnętrzną instrumentacją

- Możemy wymuszać wartości graniczne w sposób automatyczny.
- Łoża igłowe.
- Np. symulowanie uszkodzonej transmisji.
-
-

Testowanie hardwareu



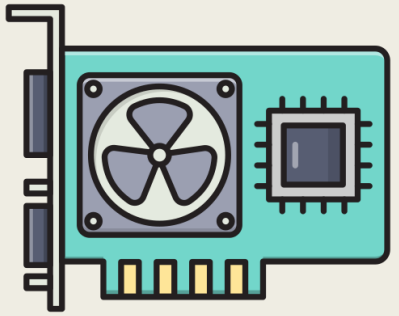
Testy automatyczne

Testy pół automatyczne

Automatyczne z zewnętrzną instrumentacją

- Możemy wymuszać wartości graniczne w sposób automatyczny.
- Łoża igłowe.
- Np. symulowanie uszkodzonej transmisji.
- Zaniki zasilania.
-

Testowanie hardwareu



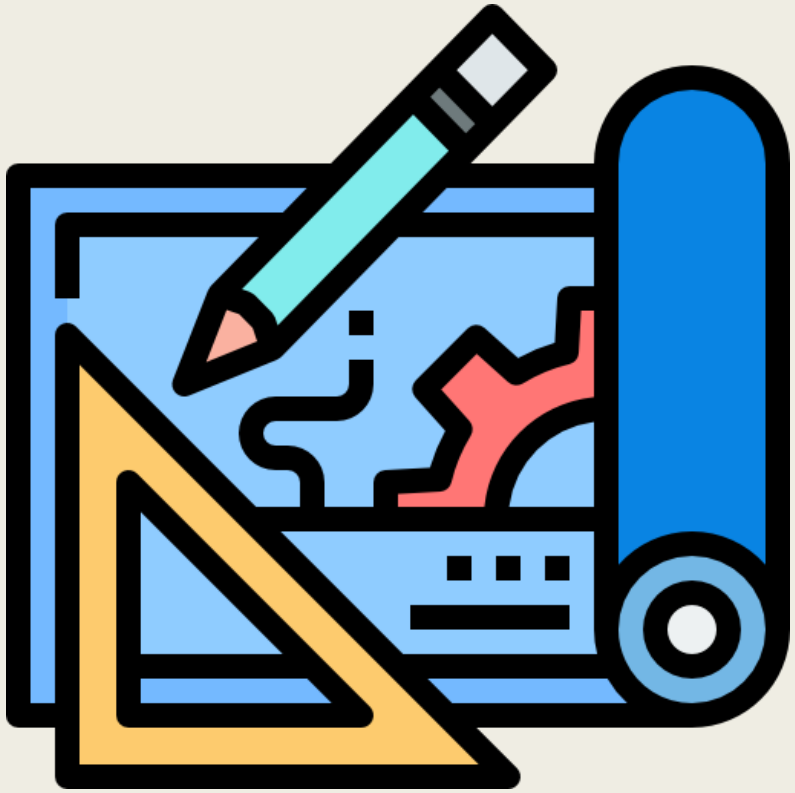
Testy automatyczne

Testy pół automatyczne

Automatyczne z zewnętrzną instrumentacją

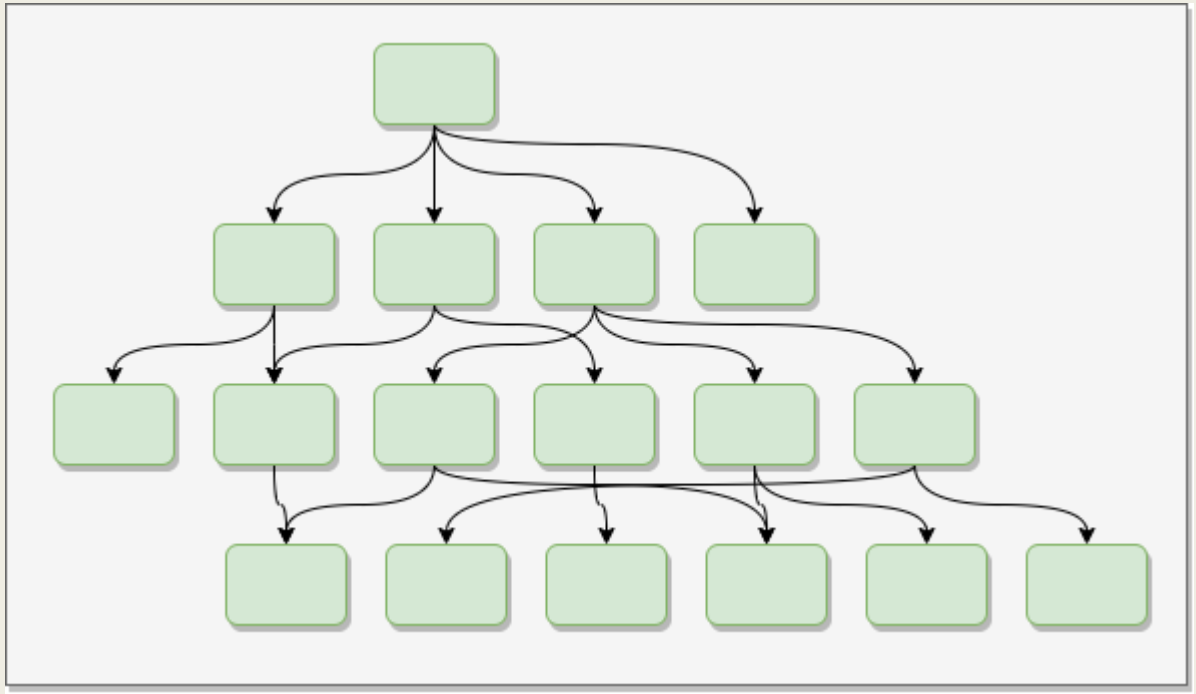
- Możemy wymuszać wartości graniczne w sposób automatyczny.
- Łoża igłowe.
- Np. symulowanie uszkodzonej transmisji.
- Zaniki zasilania.
- Zewnętrzne generatory sygnałów.

TDD i architektura

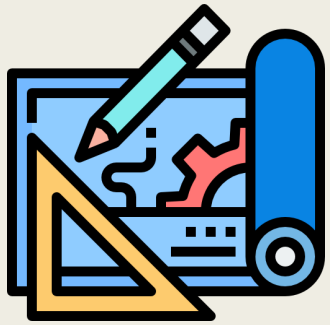


TDD i architektura

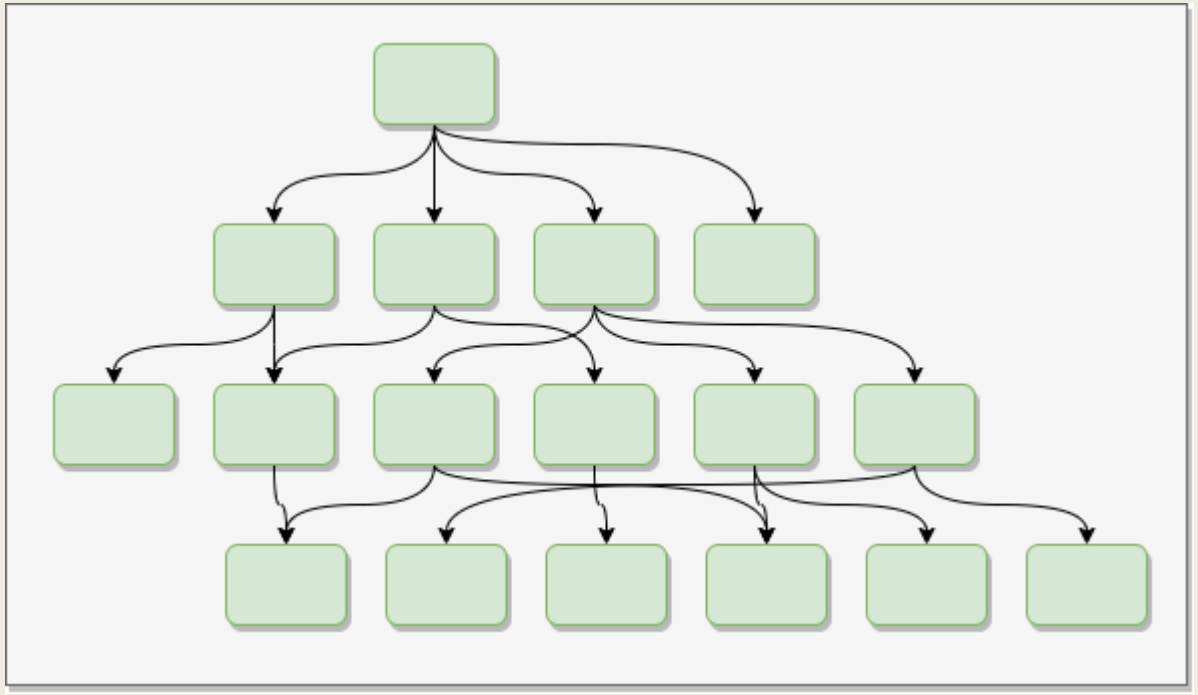
Typowa architektura.



TDD i architektura

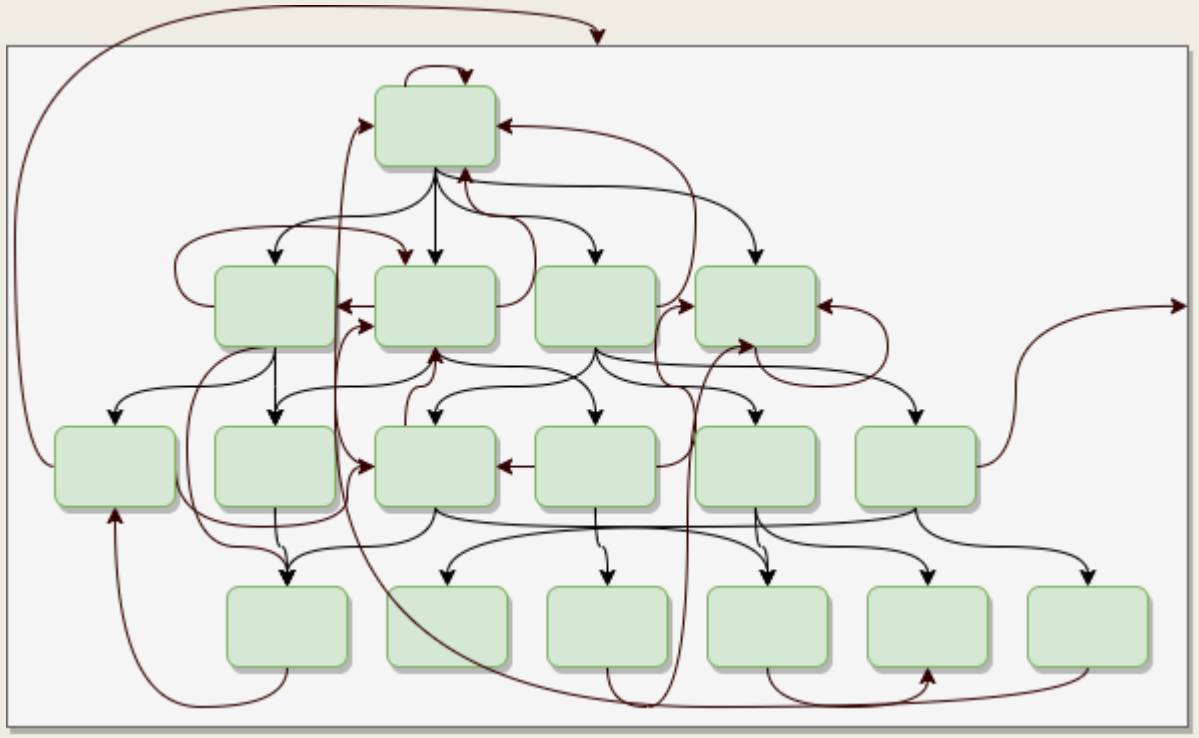


...



TDD i architektura

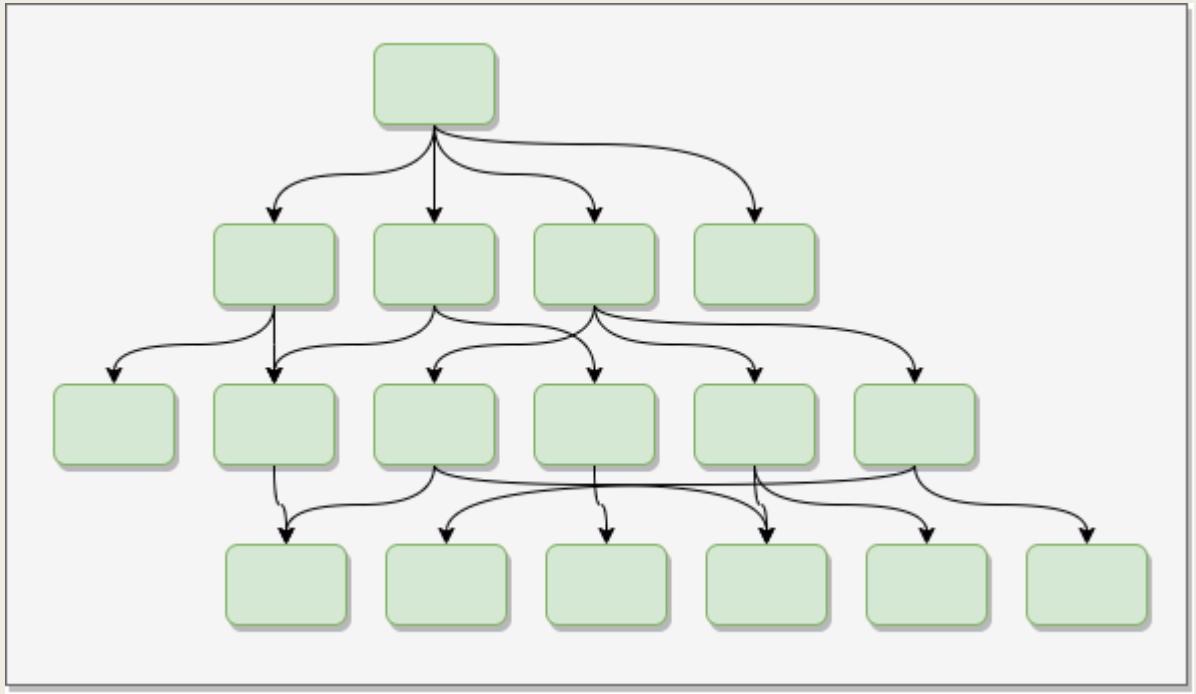
Typowa architektura.



TDD i architektura

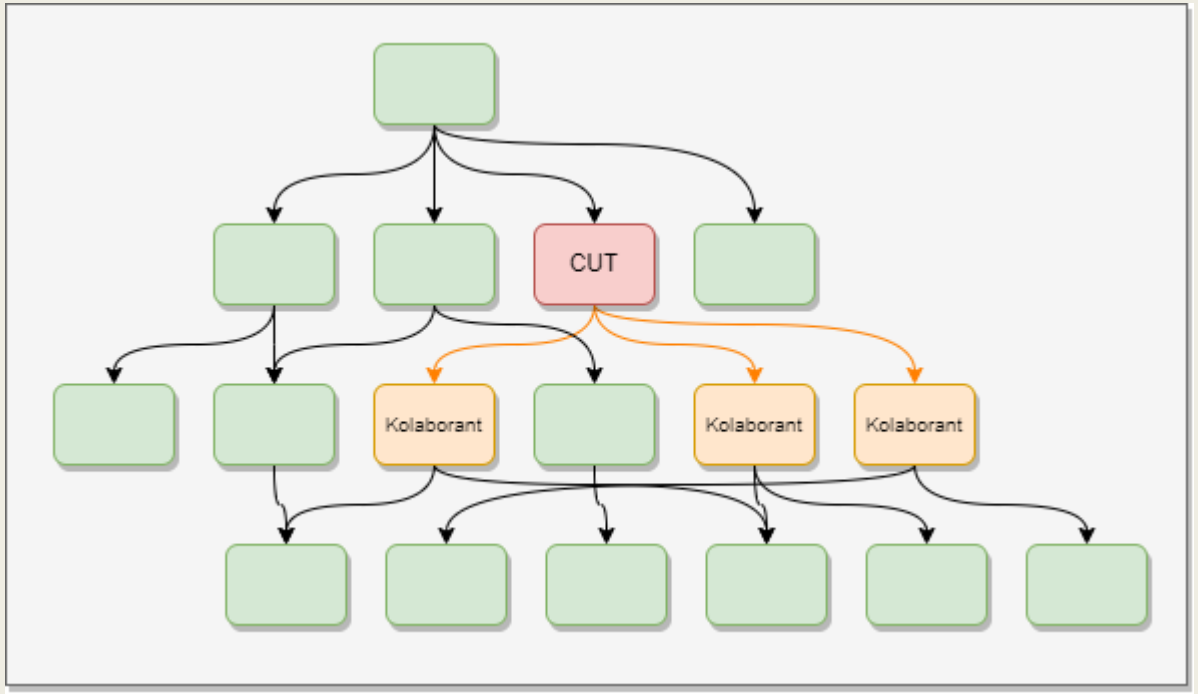
~~Typowa architektura.~~

Typowa architektura tworzona w metodologii TDD.



TDD i architektura

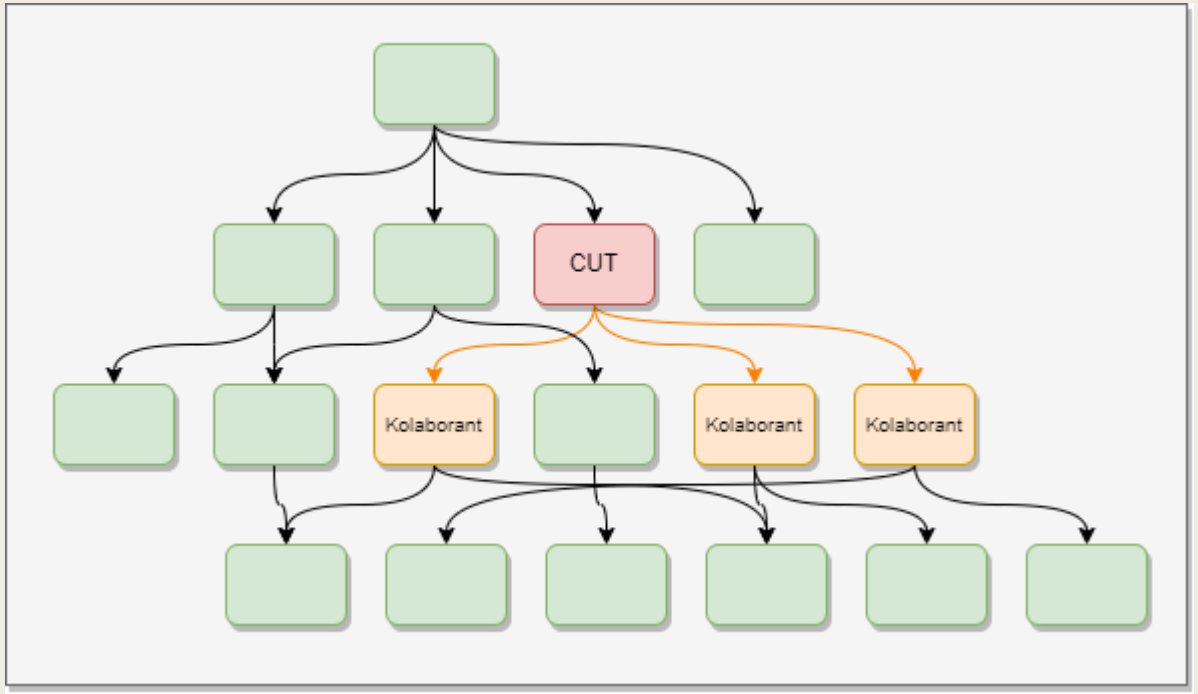
Moduły mają zależności



TDD i architektura

Moduły mają zależności

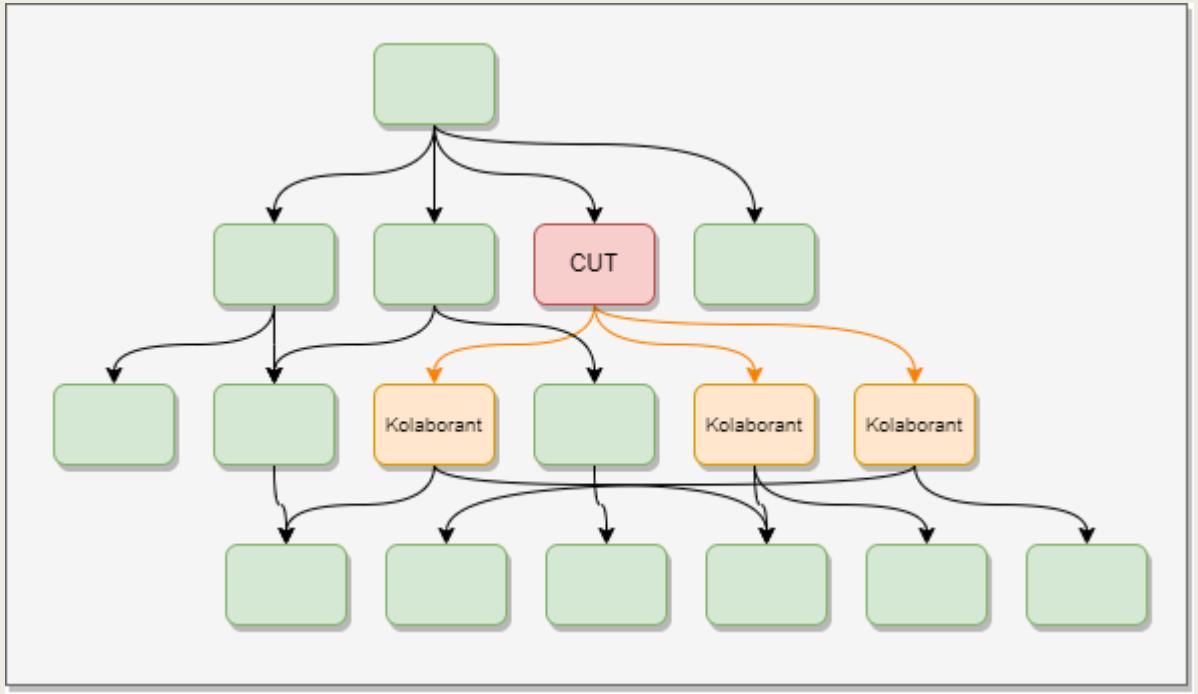
- Inne moduły
-
-
-



TDD i architektura

Moduły mają zależności

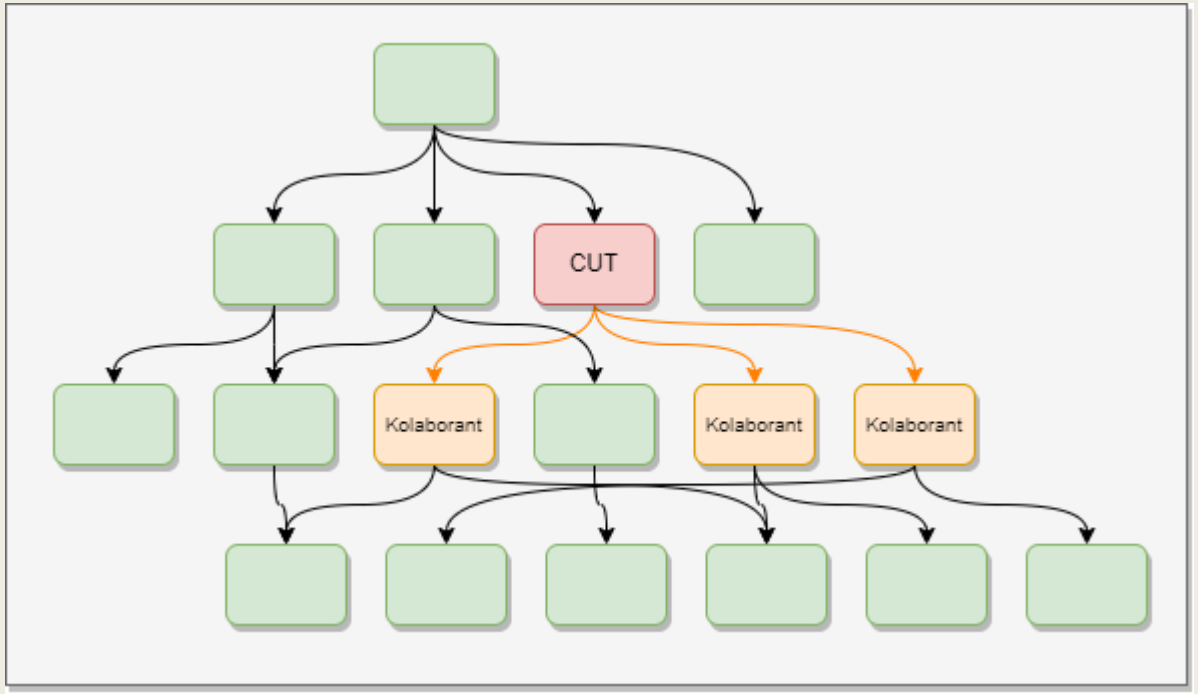
- Inne moduły
- Podłączony HW
-
-



TDD i architektura

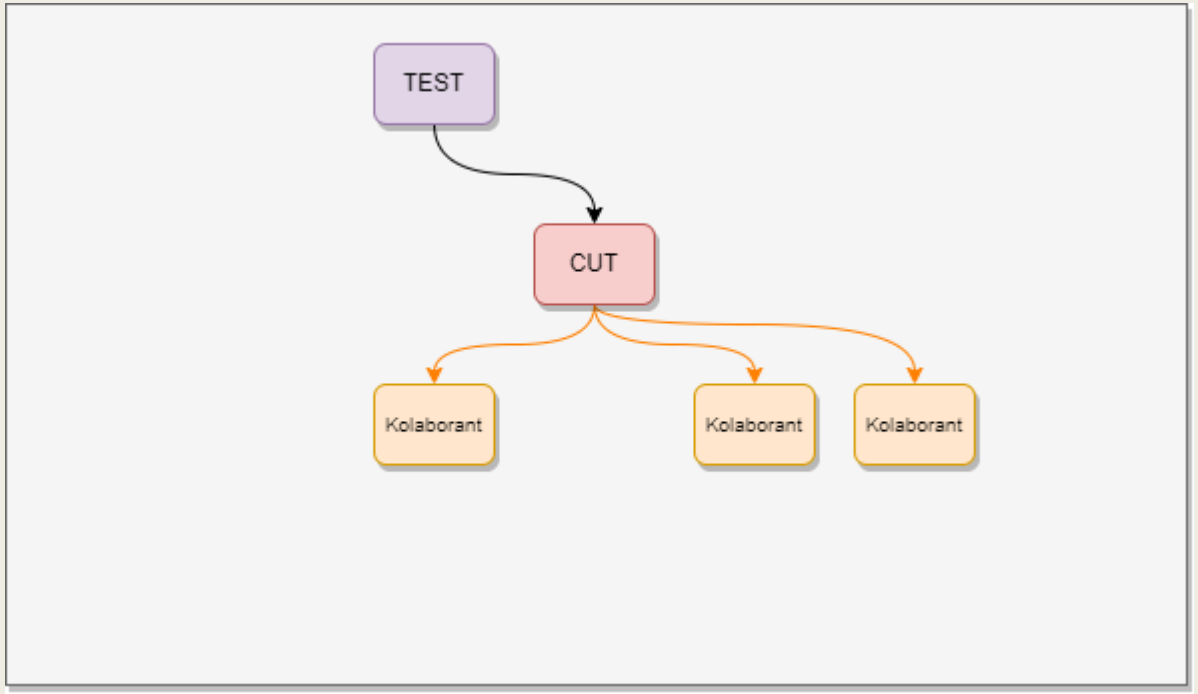
Moduły mają zależności

- Inne moduły
- Podłączony HW
- System operacyjny
- itd.



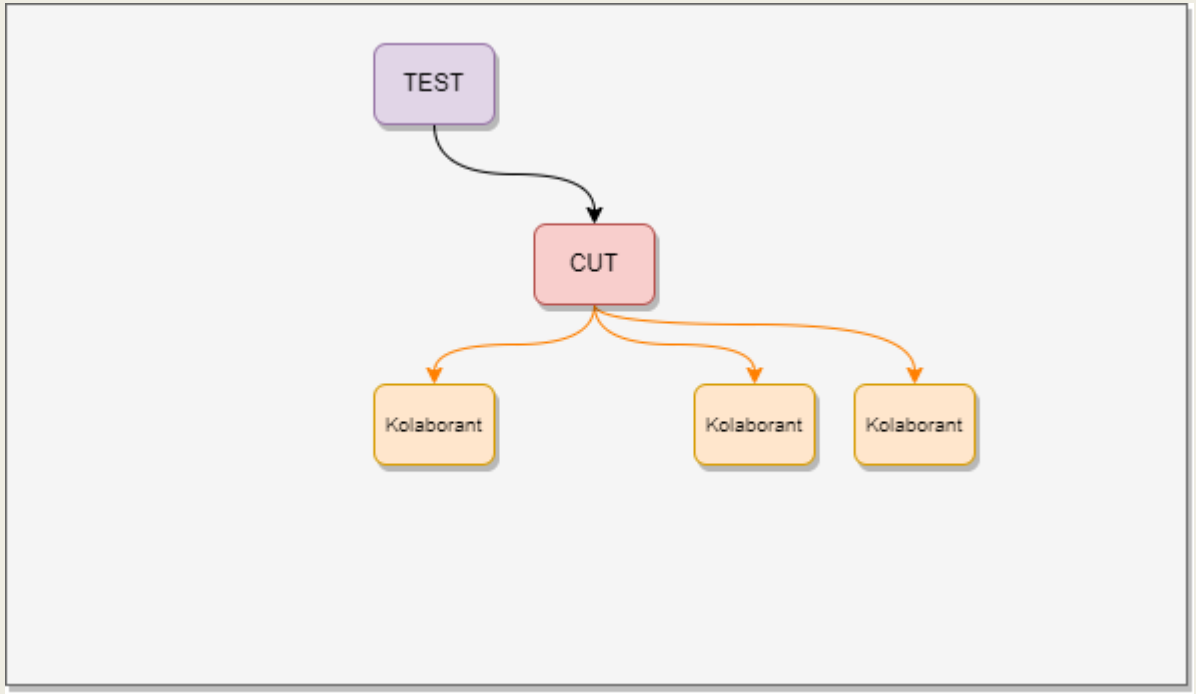
TDD i architektura

Ale powinny być testowane osobno.



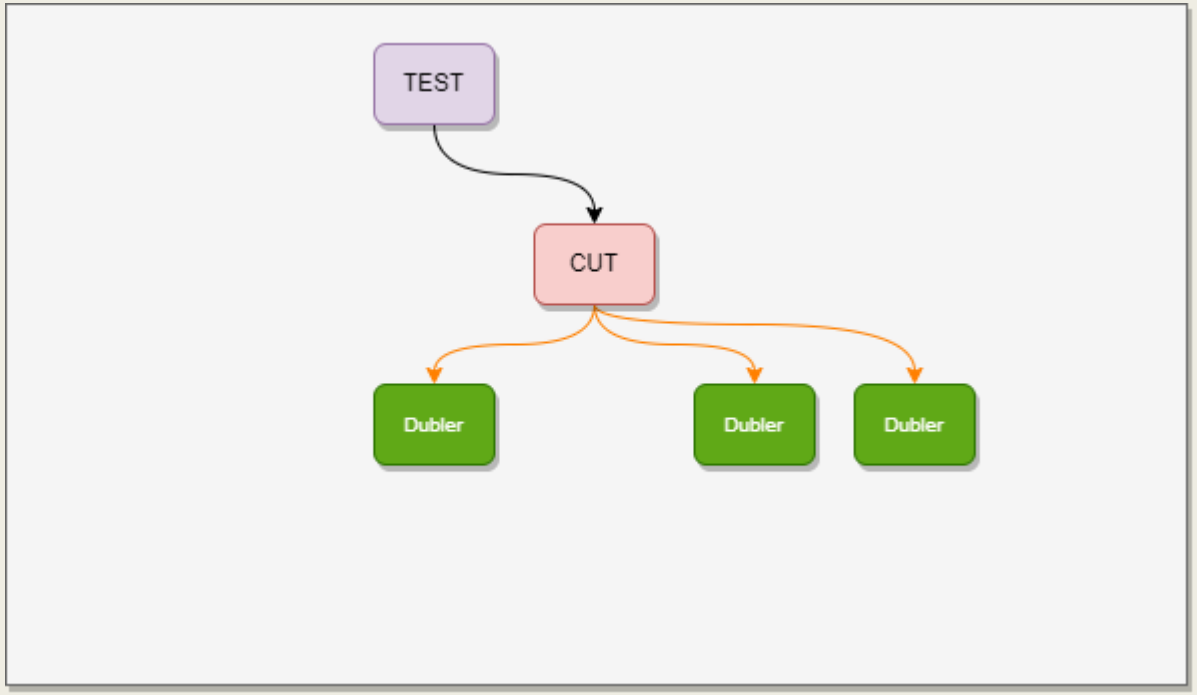
TDD i architektura

Co możemy zrobić?



TDD i architektura

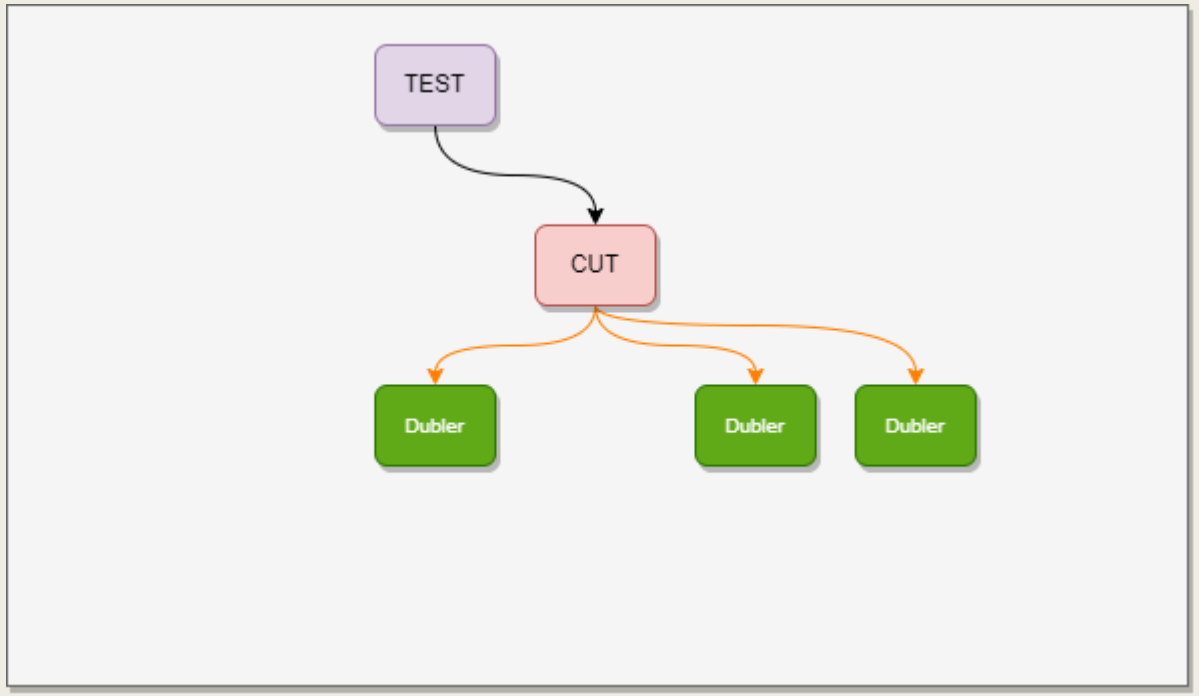
Wprowadzić dublerów.



TDD i architektura

Jakie problemy rozwiązują dublerzy?

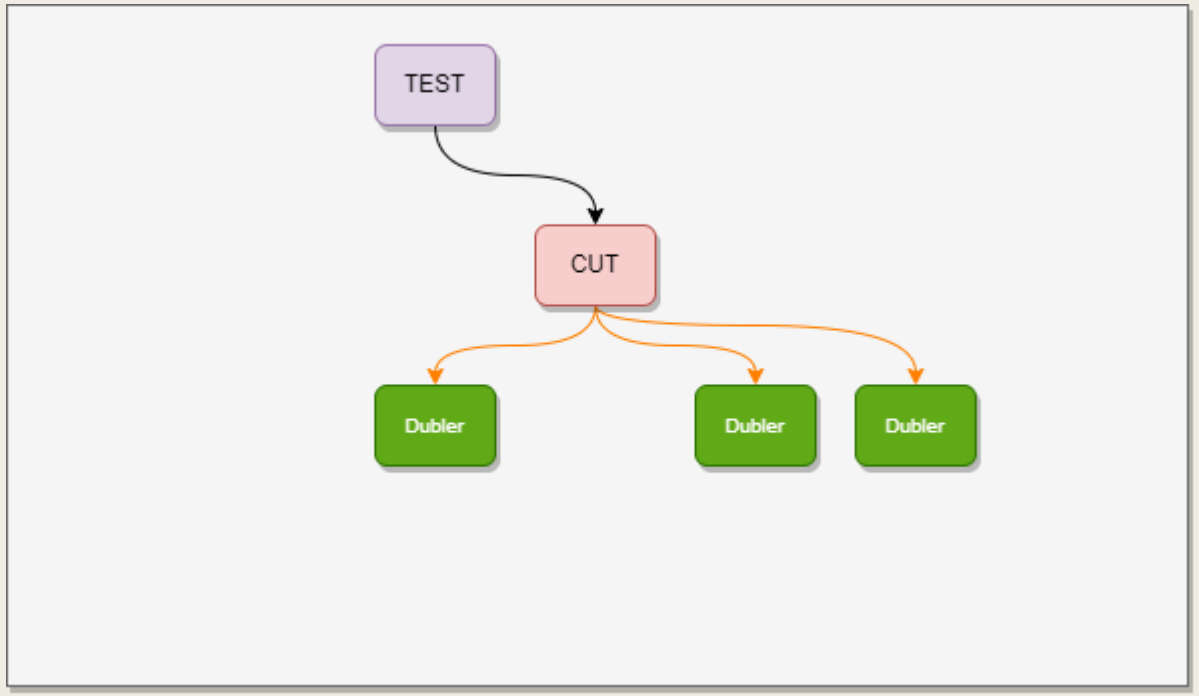
- Separacja od HW.
-
-
-
-
-



TDD i architektura

Jakie problemy rozwiązują dublerzy?

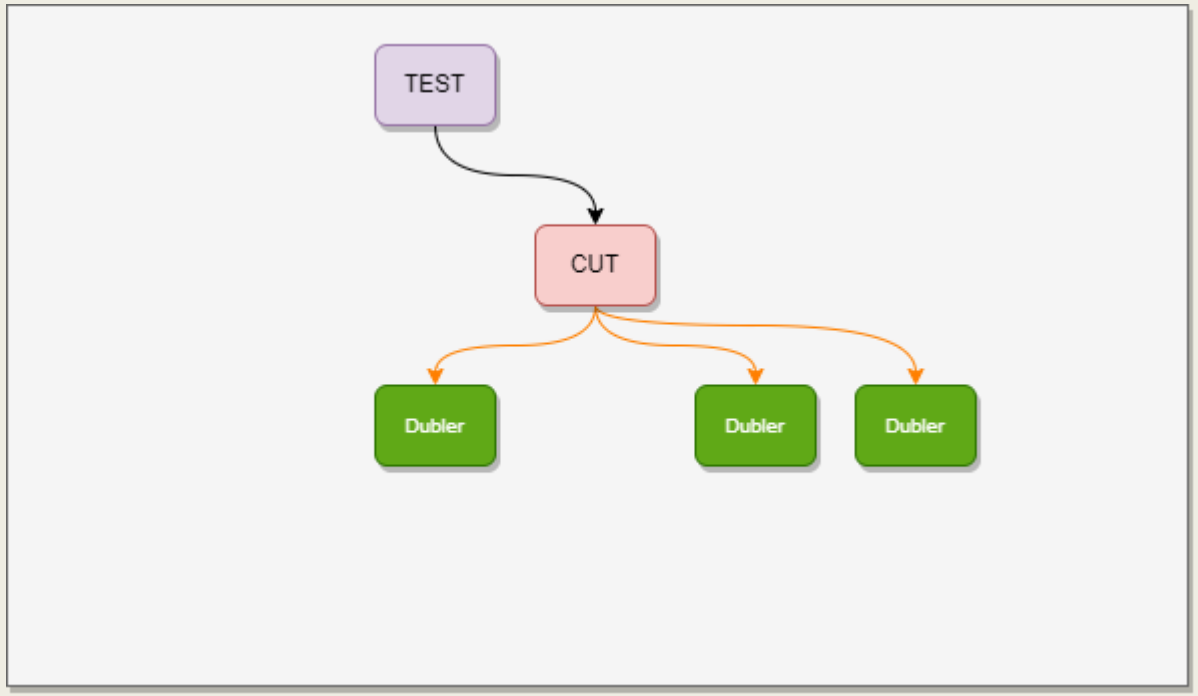
- Separacja od HW.
- Generują trudny input (np. GPS)
-
-
-
-



TDD i architektura

Jakie problemy rozwiązują dublerzy?

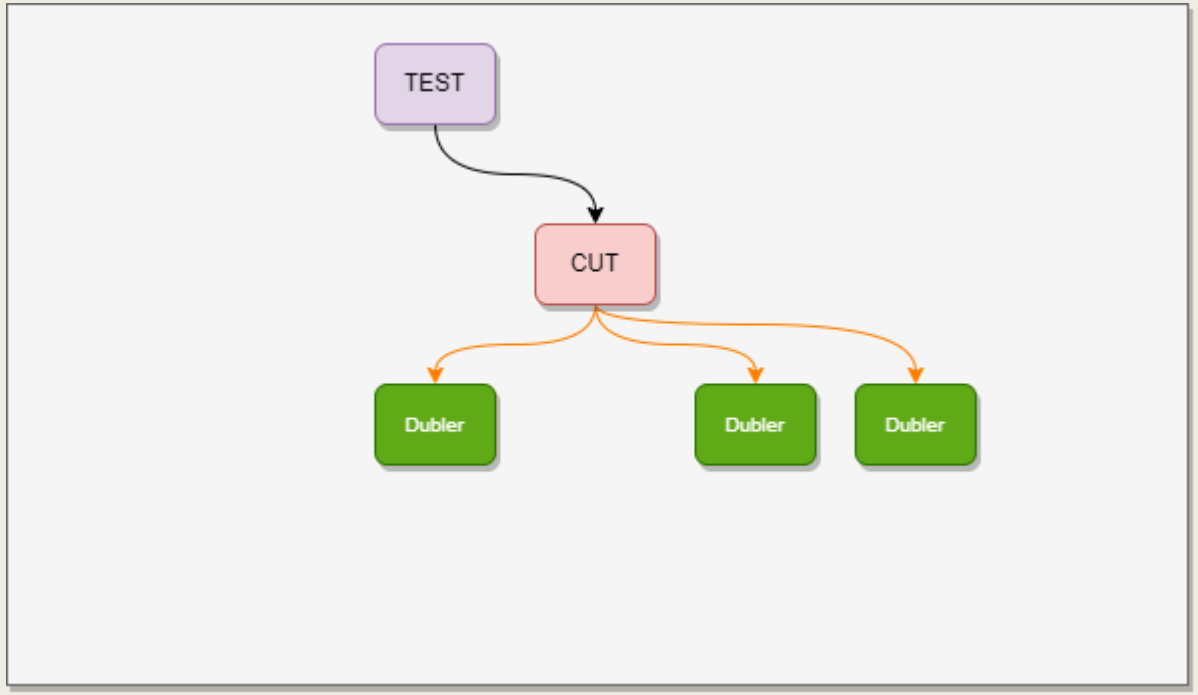
- Separacja od HW.
- Generują trudny input (np. GPS)
- Zastępują powolne moduły (np. komunikację z bazą)
-
-
-



TDD i architektura

Jakie problemy rozwiązują dublerzy?

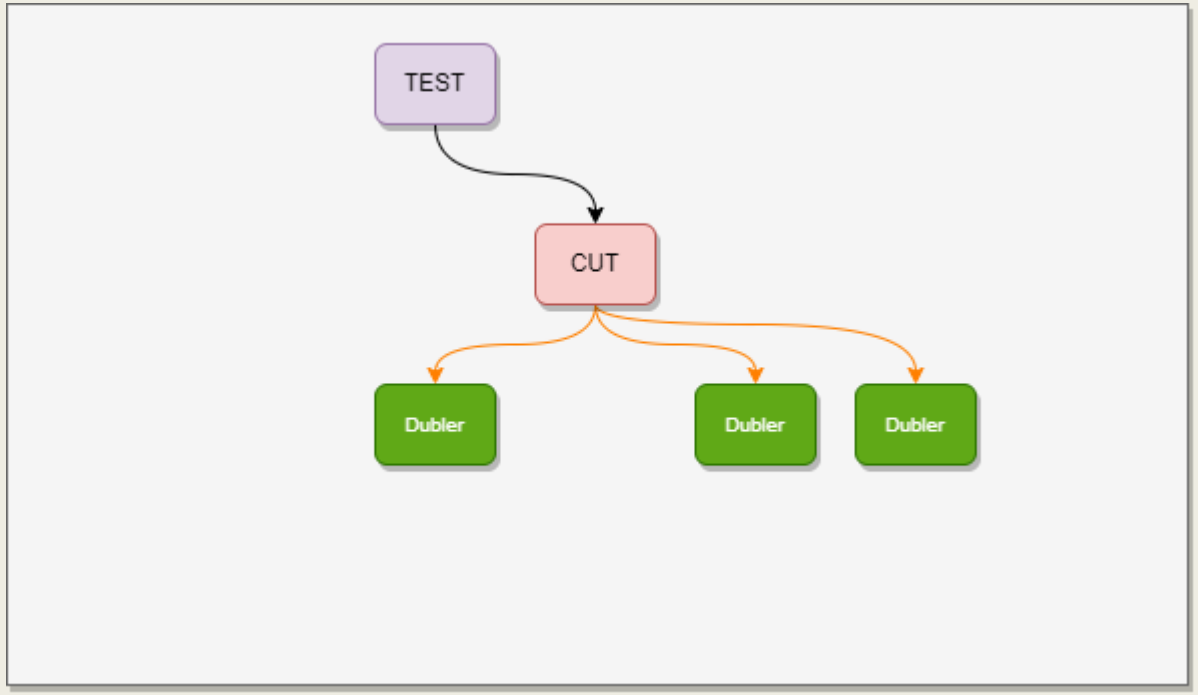
- Separacja od HW.
- Generują trudny input (np. GPS)
- Zastępują powolne moduły (np. komunikację z bazą)
- Zastępują moduły dające chwilowe wartości (np. HW RTC)
-
-



TDD i architektura

Jakie problemy rozwiązują dublerzy?

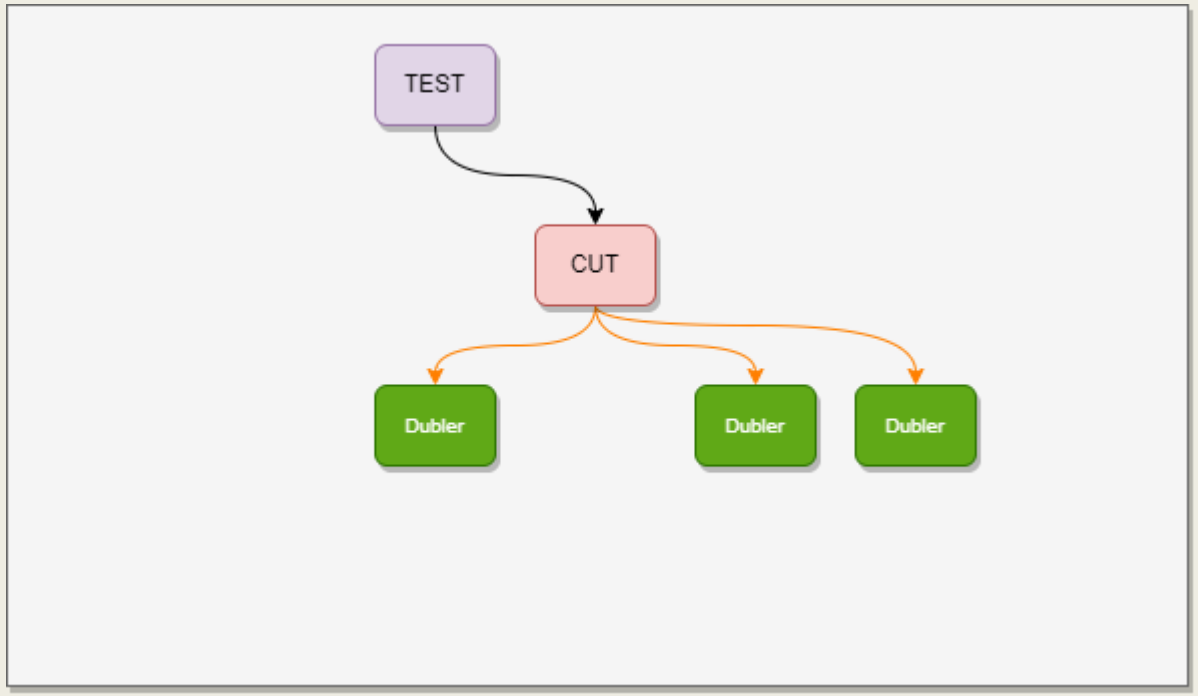
- Separacja od HW.
- Generują trudny input (np. GPS)
- Zastępują powolne moduły (np. komunikację z bazą)
- Zastępują moduły dające chwilowe wartości (np. HW RTC)
- Niegotowe moduły (mamy API)
-



TDD i architektura

Jakie problemy rozwiązują dublerzy?

- Separacja od HW.
- Generują trudny input (np. GPS)
- Zastępują powolne moduły (np. komunikację z bazą)
- Zastępują moduły dające chwilowe wartości (np. HW RTC)
- Niegotowe moduły (mamy API)
- Moduły trudne do konfiguracji



TDD i architektura

Jak podmienić kolaborantów na dublerów?

TDD i architektura

??? ??? ???

Podmiana kolaborantów.

TDD i architektura

Podczas linkowania

???

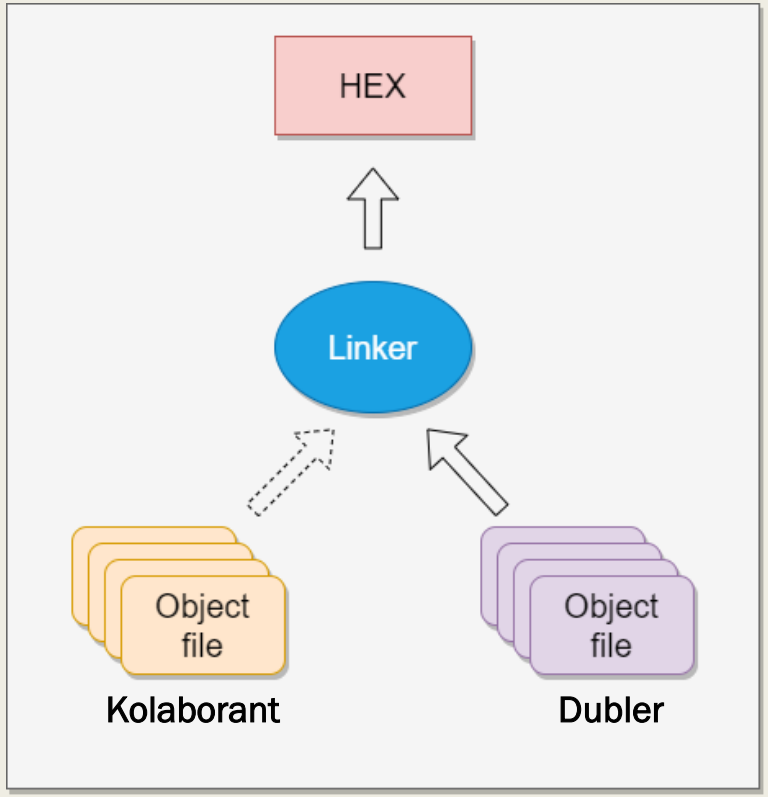
???

Podmiana kolaborantów.

TDD i architektura

Substytucja podczas linkowania

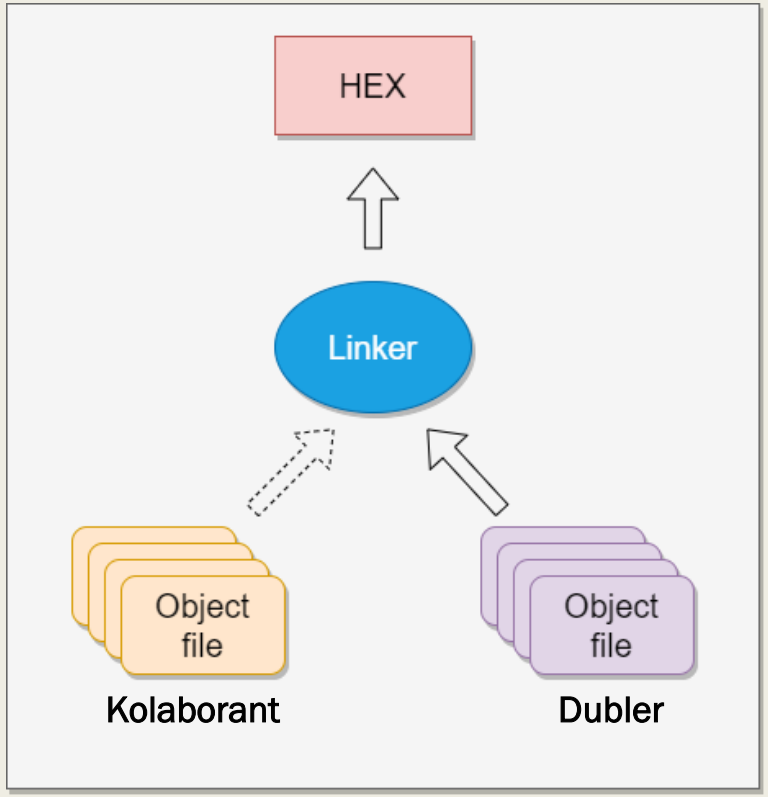
- Podmieniamy pliki **.o**
-
-
-



TDD i architektura

Substytucja podczas linkowania

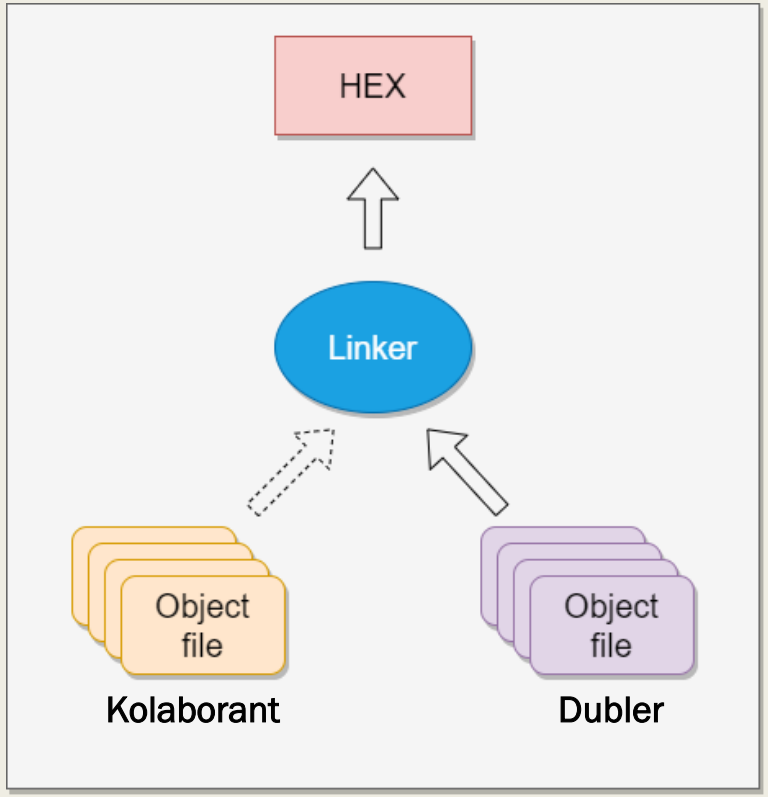
- Podmieniamy pliki **.o**
- Możemy dodać funkcje ekstra (szpiegowanie).
-
-



TDD i architektura

Substytucja podczas linkowania

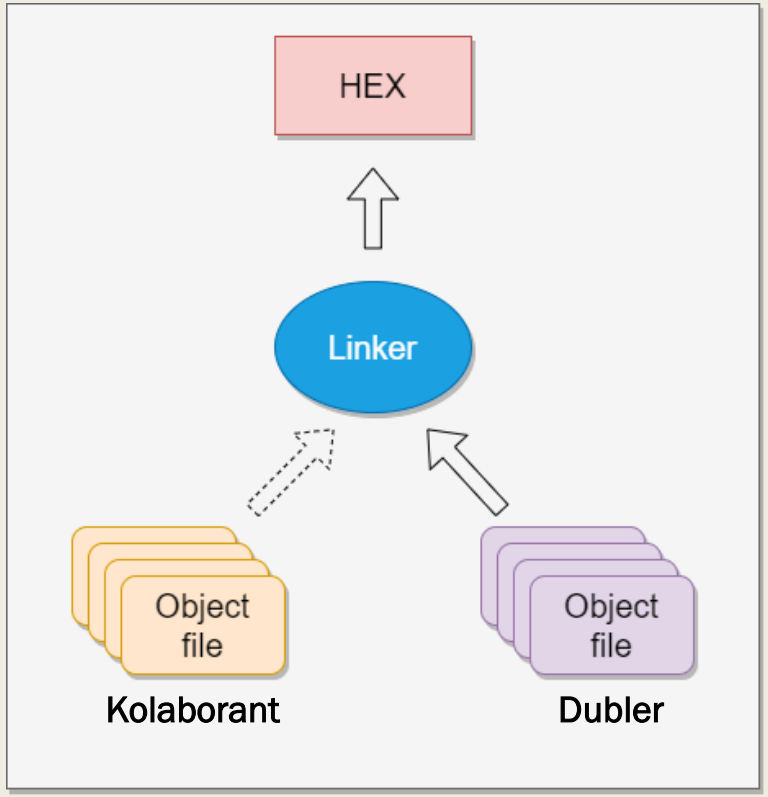
- Podmieniamy pliki **.o**
- Możemy dodać funkcje ekstra (szpiegowanie).
- Najbardziej elegancki sposób.
-



TDD i architektura

Substytucja podczas linkowania

- Podmieniamy pliki **.o**
- Możemy dodać funkcje ekstra (szpiegowanie).
- Najbardziej elegancki sposób.
- Dla testowanego modułu nic się nie zmieniło.



TDD i architektura

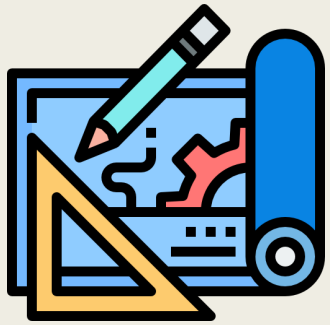
Podczas linkowania

???

???

Podmiana kolaborantów.

TDD i architektura



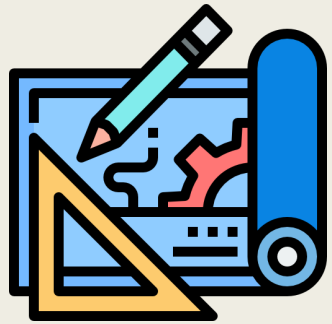
Podczas linkowania

Podmiana wskaźnika do funkcji

???

Podmiana kolaborantów.

TDD i architektura



Substytucja wskaźnika

- Mniej idealna niż podmiana podczas linkowania (ale można łączyć!)
-
-

TDD i architektura

Substytucja wskaźnika

- Mniej idealna niż podmiana podczas linkowania (ale można łączyć!)
- Wymaga trochę więcej RAMu.
-

TDD i architektura

Substytucja wskaźnika

- Mniej idealna niż podmiana podczas linkowania (ale można łączyć!)
- Wymaga trochę więcej RAMu.
- Pozwala na łatwą podmianę np. drivera w runtime.


TALK IS CHEAP
SHOW ME THE CODE
imgflip.com

TDD i architektura

Substytucja wskaźnika

To jest nasz CUT

```
void measureEnvParams(const MeasurementsApi_t* api) {
    api->readTemperature();
    api->readHumidity();
    api->readPressure();
}
```

TDD i architektura

Substytucja wskaźnika

To jest nasz CUT

```
void measureEnvParams(const MeasurementsApi_t* api) {
    api->readTemperature();
    api->readHumidity();
    api->readPressure();
}
```

Tu następuje substytucja

TDD i architektura

Substytucja wskaźnika

To jest nasz CUT

```
MeasurementsApi_t api = {
    .readTemperature = &Driver_ReadTemperature,
    .readHumidity    = &Driver_ReadHumidity,
    .readPressure    = &Driver_ReadPressure
}
```

Możemy wsadzić to

```
void measureEnvParams(const MeasurementsApi_t* api) {
    api->readTemperature();
    api->readHumidity();
    api->readPressure();
}
```

TDD i architektura

Substytucja wskaźnika

To jest nasz CUT

```
MeasurementsApi_t api = {
    .readTemperature = &MOCK_ReadTemperature,
    .readHumidity    = &MOCK_ReadHumidity,
    .readPressure    = &MOCK_ReadPressure
}
```

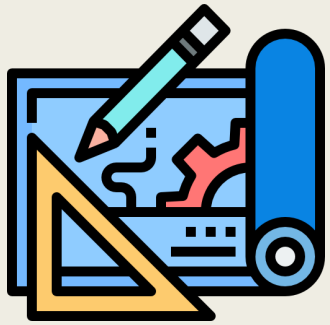
Albo to

```
MeasurementsApi_t api = {
    .readTemperature = &Driver_ReadTemperature,
    .readHumidity    = &Driver_ReadHumidity,
    .readPressure    = &Driver_ReadPressure
}
```

Możemy wsadzić to

```
void measureEnvParams(const MeasurementsApi_t* api) {
    api->readTemperature();
    api->readHumidity();
    api->readPressure();
}
```

TDD i architektura



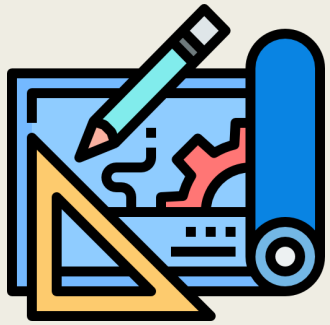
Podczas linkowania

Podmiana wskaźnika do funkcji

???

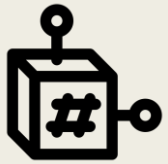
Podmiana kolaborantów.

TDD i architektura



Podczas linkowania

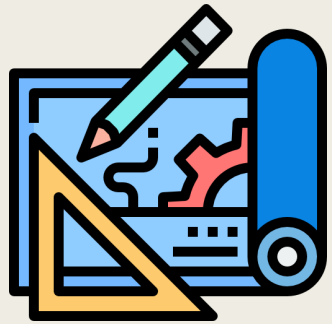
Podmiana wskaźnika do funkcji



Użycie preprocesora

Podmiana kolaborantów.

TDD i architektura



Użycie preprocesora

- Narzędzie ostatniej szansy.
-
-

TDD i architektura

Użycie preprocesora

- Narzędzie ostatniej szansy.
- Inny kod wykonywany podczas testów a inny podczas działania programu.
-

TDD i architektura

Użycie preprocesora

- Narzędzie ostatniej szansy.
- Inny kod wykonywany podczas testów a inny podczas działania programu.
- Programowanie makrami psuje czytelność i utrudnia śledzenie programu.

TDD i architektura

Użycie preprocesora

Przykład zastosowania

TDD i architektura

Użycie preprocesora

Przykład zastosowania (wycięte z cpputest):

Przygotowujemy funkcje do podmianki:

```
void* cpputest_malloc(size_t, const char*, int);
void* cpputest_calloc(size_t count, size_t size, const char*, int);
void* cpputest_realloc(void*, size_t, const char*, int);
void cpputest_free(void* mem, const char*, int);
```

TDD i architektura

Użycie preprocesora

Przykład zastosowania (wycięte z cpputest):

Przygotowujemy funkcje do podmianki:

```
void* cpputest_malloc(size_t, const char*, int);
void* cpputest_calloc(size_t count, size_t size, const char*, int);
void* cpputest_realloc(void*, size_t, const char*, int);
void cpputest_free(void* mem, const char*, int);
```

I tak podmieniamy:

```
#define malloc(a) cpputest_malloc(a, __FILE__, __LINE__)
#define calloc(a, b) cpputest_calloc(a, b, __FILE__, __LINE__)
#define realloc(a, b) cpputest_realloc(a, b, __FILE__, __LINE__)
#define free(a) cpputest_free(a, __FILE__, __LINE__)
```

Legacy code

Legacy code

- Główna broń – testy jednostkowe
-
-
-
-
-
-

Legacy code

- Główna broń – testy jednostkowe
- Problem NIH
-
-
-
-
-

Legacy code

- Główna broń – testy jednostkowe
- Problem NIH
- Mój kod pachnie
-
-
-
-

Legacy code

- Główna broń – testy jednostkowe
- Problem NIH
- Mój kod pachnie
- Czyjś kod pachnie …
-
-
-


Jego skarpetami
lostsockscorporation

Legacy code

- Główna broń – testy jednostkowe
- Problem NIH
- Mój kod pachnie
- Czyjś kod pachnie …
- Czasem jednak nagle okazuje się …
-
-


lets see who made this
idiotic program 1 year ago

Legacy code

- Główna broń – testy jednostkowe
- Problem NIH
- Mój kod pachnie
- Czyjś kod pachnie …
- Czasem jednak, to my jesteśmy autorami zapuszczonego kodu.
- Z czasem wszystko ulega entropii
-

Legacy code

- Główna broń – testy jednostkowe
- Problem NIH
- Mój kod pachnie
- Czyjś kod pachnie …
- Czasem jednak, to my jesteśmy autorami zapuszczonego kodu.
- Z czasem wszystko ulega entropii
- Każdy prędzej czy później będzie miał do czynienia ze starym kodem.

Legacy code

- Główna broń – testy jednostkowe
- Problem NIH
- Mój kod pachnie
- Czyjś kod pachnie …
- Czasem jednak, to my jesteśmy autorami zapuszczonego kodu.
- Z czasem wszystko ulega entropii
- Każdy prędzej czy później będzie miał do czynienia ze starym kodem. (**swoim lub kogoś innego**)

Legacy code

Plan bitwy

Legacy code

??? ??? ???

Plan bitwy

Legacy code

Rozpoznanie

???

???

Plan bitwy

Legacy code – plan bitwy

Rozpoznanie

- Co robi kod?
  -
  -
-
  -
  -

Legacy code – plan bitwy

Rozpoznanie

- Co robi kod?
  - *Sterowanie pompą insulinową?*
  - *Pomiary środowiskowe?*
-
  -
  -

Legacy code – plan bitwy

Rozpoznanie

- Co robi kod?
  - *Sterowanie pompą insulinową?*
  - *Pomiary środowiskowe?*
- Jaka jest jego struktura?
  - *Jak bardzo złamane są wzorce projektowe?*
  - *Jak bardzo złamane jest S.O.L.I.D.?*

Legacy code – plan bitwy

Rozpoznanie

- Prawdopodobnie zobaczymy
  - *Zduplikowany kod*
  -
  -
  -
  -
  -
  -
  -

Legacy code – plan bitwy

Rozpoznanie

- Prawdopodobnie zobaczymy
  - *Zduplikowany kod*
  - *Złe nazwy*
  -
  -
  -
  -
  -
  -

Legacy code – plan bitwy

Rozpoznanie

- Prawdopodobnie zobaczymy
  - *Zduplikowany kod*
  - *Złe nazwy*
  - *Spagetti*
  -
  -
  -
  -
  -

Legacy code – plan bitwy

Rozpoznanie

- Prawdopodobnie zobaczymy
  - *Zduplikowany kod*
  - *Złe nazwy*
  - *Spagetti*
  - *Lasagne*
  -
  -
  -
  -

Legacy code – plan bitwy

Rozpoznanie

- Prawdopodobnie zobaczymy
  - *Zduplikowany kod*
  - *Złe nazwy*
  - *Spagetti*
  - *Lasagne*
  - *Długie funkcje*
  -
  -
  -

Legacy code – plan bitwy

Rozpoznanie

- Prawdopodobnie zobaczymy
  - *Zduplikowany kod*
  - *Złe nazwy*
  - *Spagetti*
  - *Lasagne*
  - *Długie funkcje*
  - *Żenujące zagnieżdżenia (jak fale)*
  -
  -

Legacy code – plan bitwy

Rozpoznanie

- Prawdopodobnie zobaczymy
  - *Zduplikowany kod*
  - *Złe nazwy*
  - *Spagetti*
  - *Lasagne*
  - *Długie funkcje*
  - *Żenujące zagnieżdżenia*
  - *Długie listy parametrów*
  -

Legacy code – plan bitwy

Rozpoznanie

- Prawdopodobnie zobaczymy
  - *Zduplikowany kod*
  - *Złe nazwy*
  - *Spagetti*
  - *Lasagne*
  - *Długie funkcje*
  - *Żenujące zagnieżdżenia*
  - *Długie listy parametrów*
  - *Globalne zmienne*

Legacy code

Rozpoznanie

???

???

Plan bitwy

Legacy code

Rozpoznanie

Przygotowania

???

Plan bitwy

Legacy code – plan bitwy

Przygotowania

- Schowaj klawiaturę, żeby nie korciła.
-
-
-

Legacy code – plan bitwy

Przygotowania

- Schowaj klawiaturę, żeby nie korciła.
- Przypomnij sobie S.O.L.I.D.
-
-

Legacy code – plan bitwy

Przygotowania

- Schowaj klawiaturę, żeby nie korciła.
- Przypomnij sobie S.O.L.I.D.
- Rozpisz na kartce, jak naprawić kluczowe miejsca w okolicach w których będziesz przeprowadzał działania.
-

Legacy code – plan bitwy

Przygotowania

- Schowaj klawiaturę, żeby nie korciła.
- Przypomnij sobie S.O.L.I.D.
- Rozpisz na kartce, jak naprawić kluczowe miejsca w okolicach w których będziesz przeprowadzał działania.
- Podłącz klawiaturę.

Legacy code

Rozpoznanie

Przygotowania

???

Plan bitwy

Legacy code

Rozpoznanie

Przygotowania

Atak

Plan bitwy

Legacy code – plan bitwy

Atak

- Każda nowa funkcja, poprzedzona przygotowaniem testów.
-
-
-
-

Legacy code – plan bitwy

Atak

- Każda nowa funkcja, poprzedzona przygotowaniem testów.
- Napisz dublerów do podmianki.
-
-
-

Legacy code – plan bitwy

Atak

- Każda nowa funkcja, poprzedzona przygotowaniem testów.
- Napisz dublerów do podmianki.
- Przed dotknięciem istniejącej funkcji – napisz test, który będzie monitorował życie pacjenta.
-
-

Legacy code – plan bitwy

Atak

- Każda nowa funkcja, poprzedzona przygotowaniem testów.
- Napisz dublerów do podmianki.
- Przed dotknięciem istniejącej funkcji – napisz test, który będzie monitorował życie pacjenta.
- Zacznij ciąć funkcję, każdy wyekstraktowany kawałek – od razu testy.
-

Legacy code – plan bitwy

Atak

- Każda nowa funkcja, poprzedzona przygotowaniem testów.
- Napisz dublerów do podmianki.
- Przed dotknięciem istniejącej funkcji – napisz test, który będzie monitorował życie pacjenta.
- Zacznij ciąć funkcję, każdy wyekstraktowany kawałek – od razu testy.
- Pamiętaj – nie da się wygrać tej wojny jedną bitwą.

Podsumowanie

Podsumowanie

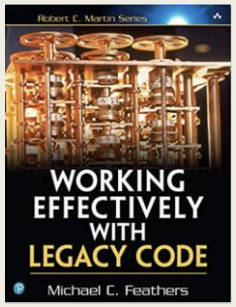
- Testy F.I.R.S.T
  - *Fast*
  - *Isolated*
  - *Repetable*
  - *Self-veryfying*
  - *Timely*
- TDD to inwestycja.
- Pomagają w poznaniu kodu i HW

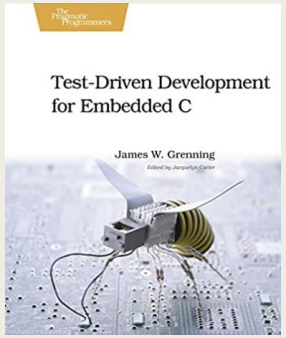
Podsumowanie

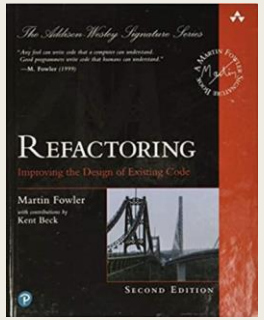
Bonus

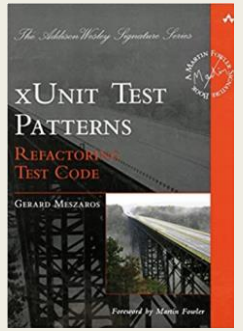
Podsumowanie

- Frameworki które można użyć i są super do embedded:
  - *CPPUTEST* - http://cpputest.github.io/
  - *Unity* - http://www.throwtheswitch.org/unity
- Książki które warto przeczytać:
  - *Michael Feathers* - ***Working Effectively with Legacy Code***
  - *Martin Fowler, with Kent Beck* - ***Refactoring, Improving the Design of Existing Code***
  - *James W. Grenning,* ***Test Driven Development for Embedded C***
  - *Gerard Meszaros,* ***xUnit Test Patterns: Refactoring Test Code***










TDD
THIS IS THE WAY.
imgflip.com

Atrybucje

- https://www.flaticon.com/authors/taufik-ramadhan
- https://www.flaticon.com/authors/vectorsmarket15
- https://www.flaticon.com/authors/uniconlabs
- https://www.flaticon.com/authors/srip
- https://www.flaticon.com/authors/vector-stall
- https://www.flaticon.com/authors/smashingstocks
- https://www.flaticon.com/authors/justicon
- https://www.flaticon.com/authors/juicy-fish
- https://www.flaticon.com/authors/orvipixel
- https://www.flaticon.com/authors/eucalyp
- https://www.flaticon.com/authors/flat-icons-design
- https://www.flaticon.com/authors/aranagraphics
- team voyas on Unsplash
- https://www.freepik.com/