Ariane

Ariane

04.06.1996 rakieta *Ariane 5G*, skonstruowana przez Europejską Agencję Kosmiczną, zeszła z kursu i została zniszczona 37 sekund po starcie. Był to pierwszy lot rakiety tego typu.

Ariane

Prace nad rakietą rozpoczęły się w roku 1984 i pochłonęły 8 mld dolarów.

Ariane

Prace nad rakietą rozpoczęły się w roku 1984 i pochłonęły 8 mld dolarów.
Śledztwo wykazało, że błąd leżał w oprogramowaniu (użyto oprogramowania z poprzedniej wersji rakiety). 64-bitowa liczba zmiennoprzecinkowa (`double` w C++) reprezentująca prędkość poziomą rakiety była rzutowana na 16-bitową liczbę całkowitą ze znakiem (`signed short int` w C++).

Ariane

Prace nad rakietą rozpoczęły się w roku 1984 i pochłonęły 8 mld dolarów.
Śledztwo wykazało, że błąd leżał w oprogramowaniu (użyto oprogramowania z poprzedniej wersji rakiety). 64-bitowa liczba zmiennoprzecinkowa (`double` w C++) reprezentująca prędkość poziomą rakiety była rzutowana na 16-bitową liczbę całkowitą ze znakiem (`signed short int` w C++). Najwyższa wartość typu `short int` wynosi +32 767. Ponieważ wartość prędkości była wyższa, wystąpił błąd oprogramowania.

Ariane

Prace nad rakietą rozpoczęły się w roku 1984 i pochłonęły 8 mld dolarów.
Śledztwo wykazało, że błąd leżał w oprogramowaniu (użyto oprogramowania z poprzedniej wersji rakiety). 64-bitowa liczba zmiennoprzecinkowa (`double` w C++) reprezentująca prędkość poziomą rakiety była rzutowana na 16-bitową liczbę całkowitą ze znakiem (`signed short int` w C++). Najwyższa wartość typu `short int` wynosi +32 767. Ponieważ wartość prędkości była wyższa, wystąpił błąd oprogramowania.
Przypadek ten jest jednym z najdroższych błędów oprogramowania w historii.

Ariane

Prace nad rakietą rozpoczęły się w roku 1984 i pochłonęły 8 mld dolarów.
Śledztwo wykazało, że błąd leżał w oprogramowaniu (użyto oprogramowania z poprzedniej wersji rakiety). 64-bitowa liczba zmiennoprzecinkowa (`double` w C++) reprezentująca prędkość poziomą rakiety była rzutowana na 16-bitową liczbę całkowitą ze znakiem (`signed short int` w C++). Najwyższa wartość typu `short int` wynosi +32 767. Ponieważ wartość prędkości była wyższa, wystąpił błąd oprogramowania.
Przypadek ten jest jednym z najdroższych błędów oprogramowania w historii. Kolejne starty rakiet z serii Ariane nie kończyły się katastrofą z powodu błędów oprogramowania :).

Mars polar lander

Mars polar lander

Mars polar lander – bezzałogowa sonda NASA. Jej celem było pierwsze lądowanie w rejonie podbiegunowym Marsa. Wystrzelona została 03.01.1999, do Marsa dotarła 03.12.1999 i rozbiła się o jego powierzchnię.

Mars polar lander

Konstrukcja sondy kosztowała ok. 60 mln dolarów.

Mars polar lander

Konstrukcja sondy kosztowała ok. 60 mln dolarów. Najprawdopodobniej katastrofę wywołał błąd oprogramowania – silnik hamujący wyłączył się przed lądowaniem. Wibracje spowodowane przez wysunięcie nóg do lądowania zostały przez oprogramowanie zinterpretowane jako kontakt z powierzchnią Marsa. Silnik hamujący został wyłączony, mimo że sonda znajdowała się 40m nad powierzchnią planety.

Mars polar lander

Konstrukcja sondy kosztowała ok. 60 mln dolarów. Najprawdopodobniej katastrofę wywołał błąd oprogramowania – silnik hamujący wyłączył się przed lądowaniem. Wibracje spowodowane przez wysunięcie nóg do lądowania zostały przez oprogramowanie zinterpretowane jako kontakt z powierzchnią Marsa. Silnik hamujący został wyłączony, mimo że sonda znajdowała się 40m nad powierzchnią planety. Przed startem było wiadomo, że wysunięcie nóg może wywołać wibracje, które mogą być tak zinterpretowane, jednak oprogramowanie nie obsługiwało takiego przypadku.

Mars polar lander

Konstrukcja sondy kosztowała ok. 60 mln dolarów.
Najprawdopodobniej katastrofę wywołał błąd oprogramowania – silnik hamujący wyłączył się przed lądowaniem. Wibracje spowodowane przez wysunięcie nóg do lądowania zostały przez oprogramowanie zinterpretowane jako kontakt z powierzchnią Marsa. Silnik hamujący został wyłączony, mimo że sonda znajdowała się 40m nad powierzchnią planety. Przed startem było wiadomo, że wysunięcie nóg może wywołać wibracje, które mogą być tak zinterpretowane, jednak oprogramowanie nie obsługiwało takiego przypadku.
Błędy te tłumaczone są słabym zarządzaniem i finansowaniem projektu (ponoć projekt był niedofinansowany w granicach 30%).

Mars polar lander

Konstrukcja sondy kosztowała ok. 60 mln dolarów.
Najprawdopodobniej katastrofę wywołał błąd oprogramowania – silnik hamujący wyłączył się przed lądowaniem. Wibracje spowodowane przez wysunięcie nóg do lądowania zostały przez oprogramowanie zinterpretowane jako kontakt z powierzchnią Marsa. Silnik hamujący został wyłączony, mimo że sonda znajdowała się 40m nad powierzchnią planety. Przed startem było wiadomo, że wysunięcie nóg może wywołać wibracje, które mogą być tak zinterpretowane, jednak oprogramowanie nie obsługiwało takiego przypadku.
Błędy te tłumaczone są słabym zarządzaniem i finansowaniem projektu (ponoć projekt był niedofinansowany w granicach 30%).
Następca sondy *Mars polar lander*, sonda *Phoenix*, z sukcesem wylądowała na Marsie 25.05.2008.

Testowanie

Celem testowania jest:

- wykrycie błędów w oprogramowaniu
- zwiększenie zaufania użytkownika do produktu

Testowanie

Celem testowania jest:

- wykrycie błędów w oprogramowaniu
- zwiększenie zaufania użytkownika do produktu

Nie da się uniknąć wszystkich błędów w dużym projekcie.

Testowanie

Celem testowania jest:

- wykrycie błędów w oprogramowaniu
- zwiększenie zaufania użytkownika do produktu

Nie da się uniknąć wszystkich błędów w dużym projekcie. Testowanie pozwoli znaleźć większość błędów w sytuacjach typowych.

Testowanie

Celem testowania jest:

- wykrycie błędów w oprogramowaniu
- zwiększenie zaufania użytkownika do produktu

Nie da się uniknąć wszystkich błędów w dużym projekcie. Testowanie pozwoli znaleźć większość błędów w sytuacjach typowych. Im większą uwagę poświęcimy testowaniu, tym bardziej niezawodny będzie finalny system.

Testowanie

Z testowaniem łączą się następujące pojęcia:

Testowanie

Z testowaniem łączą się następujące pojęcia:

Weryfikacja (*verification*)

Testowanie zgodności systemu lub jego części ze zdefiniowanymi wymaganiami.

Testowanie

Z testowaniem łączą się następujące pojęcia:

Weryfikacja (*verification*)

Testowanie zgodności systemu lub jego części ze zdefiniowanymi wymaganiami.

Atestowanie (*validation*)

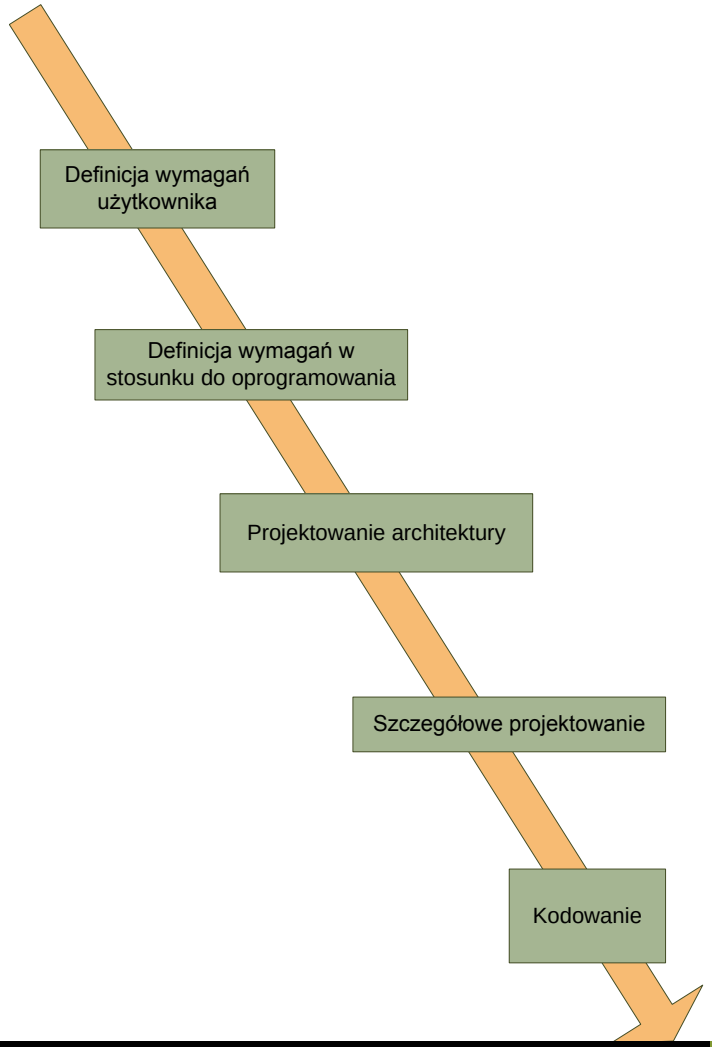
Testowanie zgodności systemu z potrzebami użytkownika.

Model *V*

Testowanie jest ściśle powiązane z poprzednimi fazami życia aplikacji.

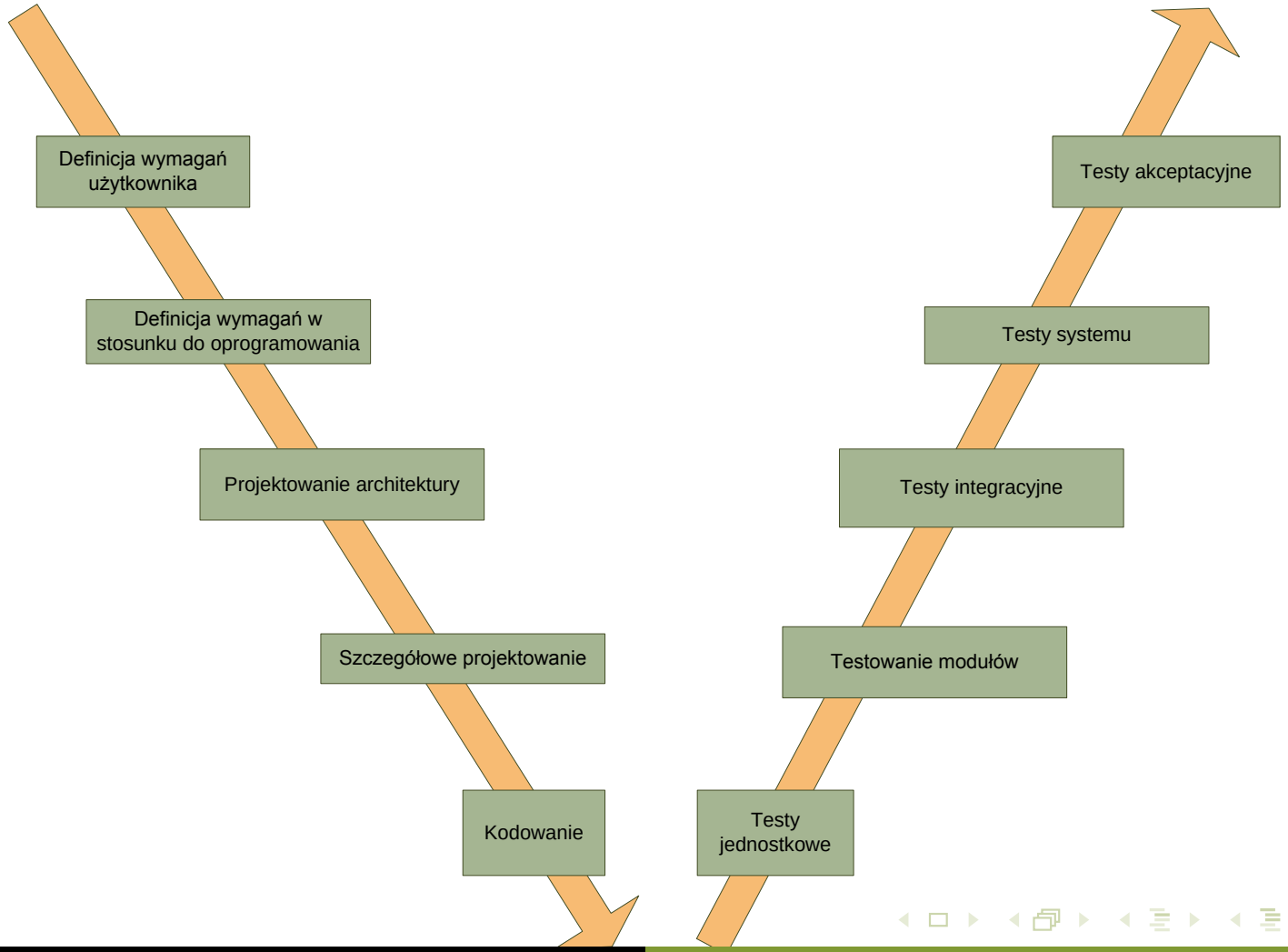
Model *V*

Testowanie jest ściśle powiązane z poprzednimi fazami życia aplikacji.



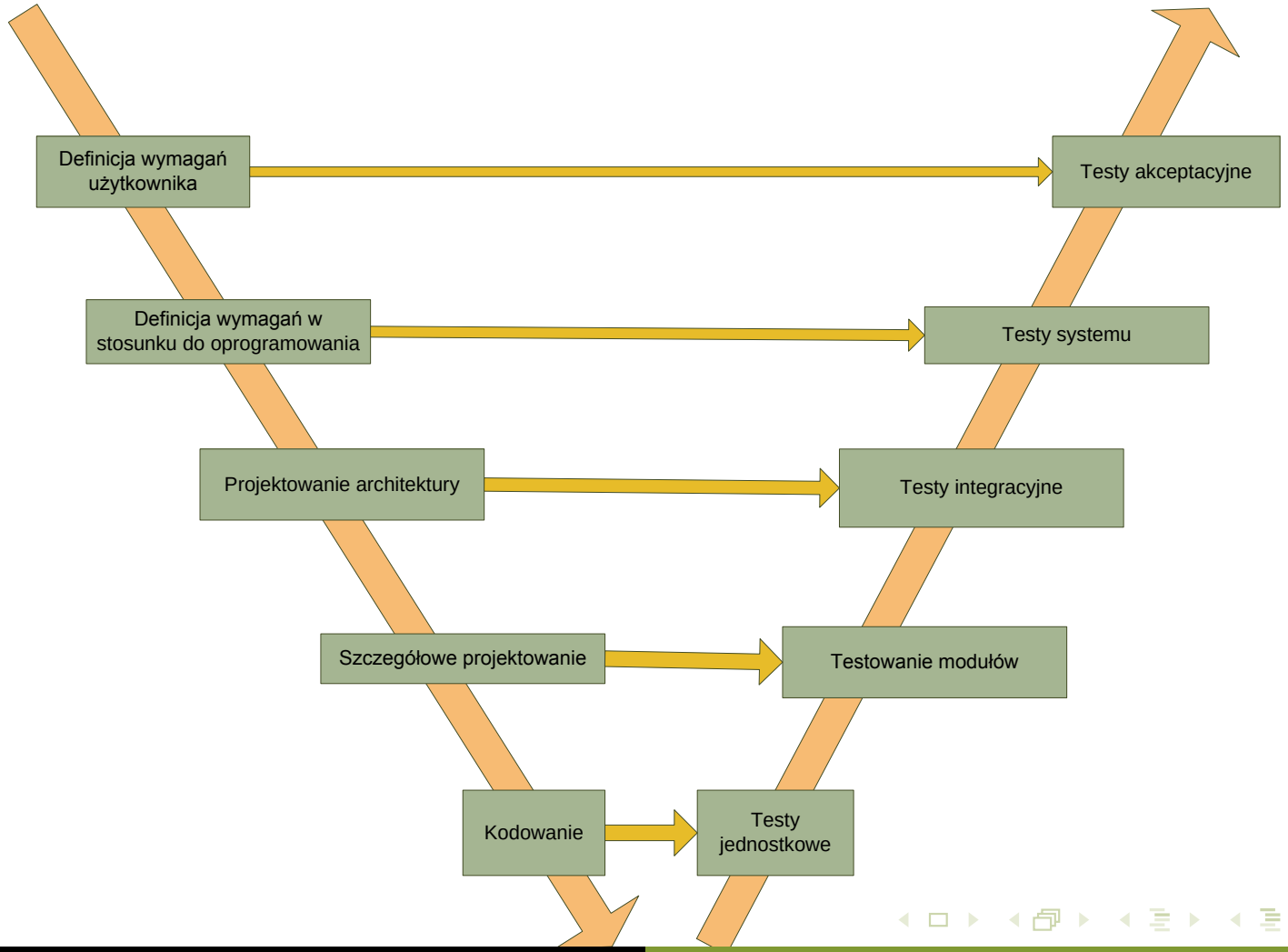
Model *V*

Testowanie jest ściśle powiązane z poprzednimi fazami życia aplikacji.



Model *V*

Testowanie jest ściśle powiązane z poprzednimi fazami życia aplikacji.



Testowanie – cele badawcze

Ponadto, w projektach o charakterze badawczym, należy dokonać testów badających ten element, np:

- czas pracy algorytmów w zależności od rozmiaru danych (np. liczba wierzchołków w grafie, liczba krawędzi w grafie)
- jakość rozwiązań znajdowanych przez algorytmu aproksymacyjne

Testowanie – cele badawcze

Ponadto, w projektach o charakterze badawczym, należy dokonać testów badających ten element, np:

- czas pracy algorytmów w zależności od rozmiaru danych (np. liczba wierzchołków w grafie, liczba krawędzi w grafie)
- jakość rozwiązań znajdowanych przez algorytmu aproksymacyjne

Typowym przedstawieniem wyników tych testów jest tabela z zestawieniem wyników.

Testowanie – cele badawcze

Człowiek trudno przyswaja tabele pełne liczb.
Wyniki testów warto przedstawiać w postaci graficznej (na wykresie) lub, gdy to niemożliwe, wygrubiać lub zaznaczać kolorem najważniejsze elementy w tabelach.

Testowanie – cele badawcze

Kilka wskazówek:

Testowanie – cele badawcze

Kilka wskazówek:

- Wykresy funkcji wykładniczych warto przedstawiać na wykresie logarytmicznym (lub logarytmiczno - logarytmicznym).

Testowanie – cele badawcze

Kilka wskazówek:

- Wykresy funkcji wykładniczych warto przedstawiać na wykresie logarytmicznym (lub logarytmiczno - logarytmicznym).
- Należy zadbać, by otrzymane wyniki były jak najbardziej porównywalne.

Testowanie – cele badawcze

Kilka wskazówek:

- Wykresy funkcji wykładniczych warto przedstawiać na wykresie logarytmicznym (lub logarytmiczno - logarytmicznym).
- Należy zadbać, by otrzymane wyniki były jak najbardziej porównywalne.
- Przy porównywaniu algorytmów warto zidentyfikować przypadki, w których algorytmy działają szczególnie dobrze lub szczególnie źle.

Testowanie – cele badawcze

Kilka wskazówek:

- Wykresy funkcji wykładniczych warto przedstawiać na wykresie logarytmicznym (lub logarytmiczno - logarytmicznym).
- Należy zadbać, by otrzymane wyniki były jak najbardziej porównywalne.
- Przy porównywaniu algorytmów warto zidentyfikować przypadki, w których algorytmy działają szczególnie dobrze lub szczególnie źle.
- Testy należy zakończyć komentarzem, zawierającym wnioski.

Testy jednostkowe

Testy jednostkowe badają poprawność pojedynczych procedur. Im bardziej podstawowa procedura (i tym samym częściej wywoływana), powinna być dokładniej przetestowana.

Testy jednostkowe

Testy jednostkowe badają poprawność pojedynczych procedur. Im bardziej podstawowa procedura (i tym samym częściej wywoływana), powinna być dokładniej przetestowana.

Test driven development

Wiodący nurt w tworzeniu oprogramowania użytkowego. W myśl TDD, najpierw przygotowujemy testy, a następnie piszemy treść procedury.

Test driven development

Test driven development

1. Stwórz zbiór testów.

Test driven development

1. Stwórz zbiór testów.
2. Wybierz test ze zbioru.

Test driven development

1. Stwórz zbiór testów.
2. Wybierz test ze zbioru.
3. Napisz tyle kodu, aby ten test przeszedł.

Test driven development

1. Stwórz zbiór testów.
2. Wybierz test ze zbioru.
3. Napisz tyle kodu, aby ten test przeszedł.
4. Jeśli test nie przechodzi, wróć do 3.

Test driven development

1. Stwórz zbiór testów.
2. Wybierz test ze zbioru.
3. Napisz tyle kodu, aby ten test przeszedł.
4. Jeśli test nie przechodzi, wróć do 3.
5. Sprawdź, czy przechodzą wszystkie poprzednie testy.

Test driven development

1. Stwórz zbiór testów.
2. Wybierz test ze zbioru.
3. Napisz tyle kodu, aby ten test przeszedł.
4. Jeśli test nie przechodzi, wróć do 3.
5. Sprawdź, czy przechodzą wszystkie poprzednie testy.
6. Jeśli nie, popraw kod i wróć do 5.

Test driven development

1. Stwórz zbiór testów.
2. Wybierz test ze zbioru.
3. Napisz tyle kodu, aby ten test przeszedł.
4. Jeśli test nie przechodzi, wróć do 3.
5. Sprawdź, czy przechodzą wszystkie poprzednie testy.
6. Jeśli nie, popraw kod i wróć do 5.
7. Jeśli są jeszcze niesprawdzone testy, wróć do 2.

Test driven development

Takie podejście ułatwia

- zrozumienie działania danej procedury
- tworzenie kodu, który robi dokładnie to, co ma robić
- tworzenie kodu niezawierającego błędów (przynajmniej w teorii)

Test driven development

Takie podejście ułatwia

- zrozumienie działania danej procedury
- tworzenie kodu, który robi dokładnie to, co ma robić
- tworzenie kodu niezawierającego błędów (przynajmniej w teorii)

Testy jednostkowe powinny obejmować różnorodne ścieżki wywołania procedury (łącznie z sytuacjami błędnymi). Wszystkie ścieżki wywołania procedury należy podzielić na pewne klasy abstrakcji (pogrupować podobne) i przygotować co najmniej jeden test dla każdej z klas.

Test driven development

Istnieje wiele narzędzi wspierających TDD:

- NUnit – .NET
- Visual Studio Team Edition – .NET
- JUnit – Java
- Jtest – Java
- PyUnit – Python