Smocza lekcja testowania oprogramowania

Kari Kakkonen / 5.02.2021
wersja 1.0

Wersja polska SJSI
Lucjan Stapp / 31.07.2021 r.
wersja 1.1 PL (25.10.2022 r.)

O prezentacji

O prezentacji (1/2)

- W tej prezentacji porównano fakty dotyczące technologii informacyjnych, kodowania i testowania oprogramowania ze światem opowieści fantasy.
- Prezentacja ta ma na celu wsparcie nauczania w szkołach, a jej przeprowadzenie zajmie 1 lub 2 lekcje.
- Niniejsza prezentacja jest objęta licencją *Creative Commons*, co oznacza, że można z niej swobodnie korzystać w celach niekomercyjnych.
- Prezentacja będzie co jakiś czas aktualizowana i jest dostępna tutaj:
  - https://www.dragonsout.com/p/presentation-for-teachers.html
- Pomysły na poprawienie można przesyłać na adres:
  - feedback@dragonsout.com
- Sprawmy, aby testowanie oprogramowania stało się znane nowym pokoleniom!

O prezentacji (2/2)

- Polska wersja tej prezentacji została przygotowana przez zespół Stowarzyszenia Jakości Systemów Informatycznych.
- Prezentacja będzie co jakiś czas aktualizowana i jest dostępna tutaj:
  - https://sjsi.org/dla-nauczycieli/
- Pomysły na poprawianie i rozwinięcie wersji polskiej można przesyłać na adres:
  - kontakt@sjsi.org
- Ułatwiajmy nowym pokoleniom zrozumienie pojęcia „testowanie oprogramowania”!

O książce „Smoki, fora ze dwora!”

- Prezentacja ta oparta jest na świecie i postaciach z książki *Smoki, fora ze dwora!*
- Książka uzupełnia proces nauczania, ale w nauczaniu nie jest wymagane korzystanie z tej prezentacji.
- Autor: Kari Kakkonen
- Ilustracje: Adrienn Széll
- Tłumaczenie: Lucjan Stapp
- Prawa do tekstu i ilustracji Dragons Out Oy
- Więcej informacji: www.dragonsout.com

O ćwiczeniach

- Ćwiczenia rysunkowe
  - Potrzebujesz papieru i ołówków.
  - Możesz zrobić zdjęcie rysunku i wgrać je m.in. do serwisu https://padlet.com
    - Nauczyciel musi stworzyć „tablicę” Padlet w celu dzielenia się lekcją.
    - To zabawna, interaktywna część lekcji.
  - Do robienia zdjęć potrzebny jest telefon komórkowy.
- Ćwiczenia testowe
  - Potrzebujesz własnego telefonu lub tabletu oraz dowolnej aplikacji, którą chcesz przetestować.
    - Możesz współdzielić telefon z przyjacielem i wykonywać ćwiczenie jako para.
- Możesz wykonywać ćwiczenia jako 5-minutowe ćwiczenia „na czas” lub możesz wykorzystać na nie tyle czasu, ile chcesz.

Smoki / defekty

Co to jest oprogramowanie?

- Oprogramowanie jest wszędzie: w grach, sklepach internetowych czy systemach sterowania samochodami, tak jak świat fantasy jest pełen zamków i wiosek.
- Oprogramowanie może również uruchomić urządzenie.
- Nazywane również programem, chociaż w rzeczywistości wiele programów tworzy oprogramowanie.

Oprogramowanie = kod komputerowy, który pozwala użytkownikowi wykonać jakąś akcję, na przykład zagrać w grę. Inaczej nazywane softwarem.

Co to jest defekt?

- Każde oprogramowanie ma defekty, ponieważ mylić się jest rzeczą ludzką.
- Defekty utrudniają korzystanie z oprogramowania, podobnie jak smoki nękają zamki i ich mieszkańców w opowieściach fantasy.
- Defekty oprogramowania są przypadkowe i należy je usunąć, najlepiej zanim użytkownik oprogramowania je znajdzie.
- Defekt często jest nazywany pluskwą.

Defekt = problem w kodzie oprogramowania, który powoduje awarię oprogramowania. Ludzie czasami nazywają go błędem, chociaż, właściwie, błąd jest działaniem ludzkim, które skutkuje defektem w kodzie.

Ćwiczenie 5-15 minut:

Zaprojektuj własny defekt – narysuj własnego smoka

// Czego potrzebujesz:

papieru i ołówka

// Zadanie

1 **Pomyśl o defekcie, z którym się spotkałeś.**
- Zapisz nazwę defektu i kilka słów, aby go opisać.

2 **Pomyśl o odpowiadającym mu smoku.**
- Zanotuj cechy smoka.
- Jeśli defekt był poważny, smok jest duży itp.

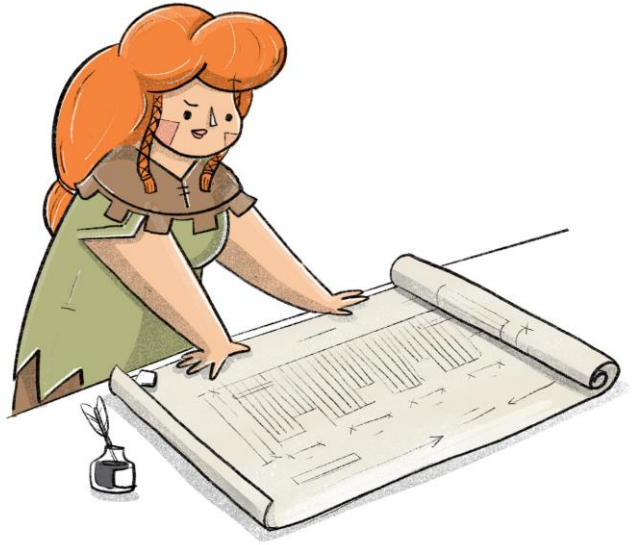
3 **Narysuj smoka.**
- Najważniejsze jest to, abyś zrealizował swój pomysł na to, jak smok może reprezentować defekt.
- Nie musisz dążyć do idealnego obrazka.

4 **Pokaż obrazek innym – jak uzgodniliście.**
- Użyj np. Padletu.

Na czym polega testowanie? Co to jest wytwarzanie oprogramowania?

Co to jest wytwarzanie oprogramowania?

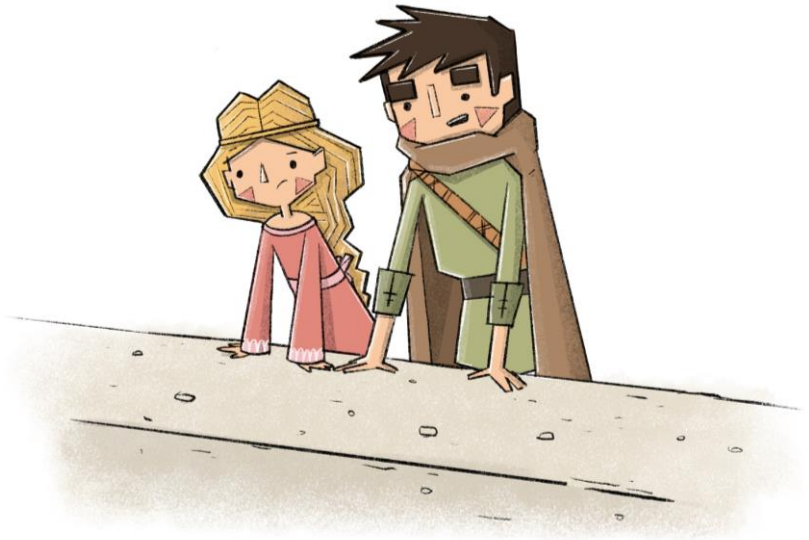
- Musisz zdecydować, co twoje oprogramowanie ma robić.
- Musisz to zakodować.
- Musisz przetestować działanie oprogramowania.
- W opowieści fantasy musisz zaprojektować i zbudować mury wokół wioski, a jednocześnie pokonać smoki, które tę wioskę trapią.



Wytwarzanie oprogramowania = wszystkie możliwe zadania wymagane do tego, by oprogramowanie działało. Obejmują one definiowanie wymagań, kodowanie i testowanie.

Co to jest testowanie?

- Musisz znaleźć defekty, aby można je było naprawić.
- Testowanie polega na szukaniu defektów i znajdowaniu ich, tak jak w opowieściach fantasy oglądasz smoki na ścianie.
- W wyniku testów czasami znajdujesz defekty.
- Testowanie w rzeczywistości nie znajduje defektu, ale znajduje awarię oprogramowania spowodowaną defektem w kodzie.



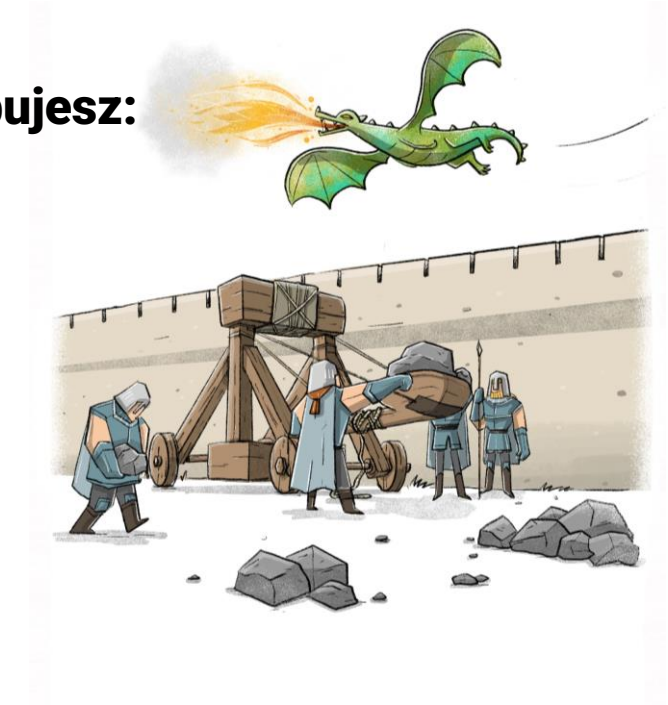
Testować = próbować, sprawdzać lub testować przy użyciu różnych wartości, w różnej kolejności, na różne sposoby, czy coś w ogóle działa lub na ile dobrze działa.

Ćwiczenie 5-15 minut:

Zaprojektuj swoje oprogramowanie – narysuj swój własny zamek

// Czego potrzebujesz:

papieru i ołówka



// Zadanie

1 **Pomyśl o oprogramowaniu, którego używałeś.**
- Zapisz nazwę oprogramowania i kilka słów, aby je opisać.
- Napisz, jak to oprogramowanie toleruje defekty.

2 **Pomyśl o odpowiadającym mu zamku, który wytrzymałby atak smoka.**
- Scharakteryzuj zamek.
- Mały czy duży? Mury obronne? Broń?

3 **Narysuj zamek.**
- Najważniejsze jest to, abyś zrealizował swój pomysł na to, jak zamek będzie obrazował oprogramowanie.
- Nie musisz dążyć do idealnego obrazka.

4 **Pokaż obrazek innym – tak, jak uzgodniliście.**
- Użyj np. Padletu.

Techniki testowania

- Testować można na wiele sposobów – są różne techniki testowania.
  - Spróbuj poprawnie korzystać z oprogramowania.
  - Spróbuj użyć oprogramowania w niewłaściwy sposób.
  - Zbadaj, jak działa oprogramowanie.
  - Obserwuj, kiedy ktoś inny korzysta z oprogramowania.
- Możesz użyć wielu technik testowania jednocześnie, tak jak w opowieści fantasy rycerz może zarówno narysować mapę, jak i zapytać ludzi, gdzie widzieli smoka.

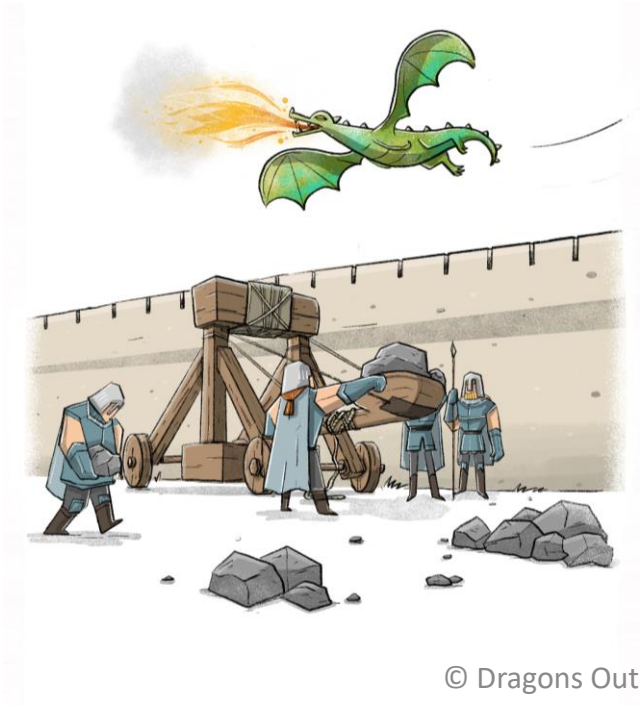
Technika testowania = sposób, w jaki wybierasz odpowiednią liczbę testów do testowania, by osiągnąć dobre pokrycie. Istnieje wiele różnych technik dostosowanych do różnych potrzeb.

Ćwiczenie 5-30 minut:

Przetestuj oprogramowanie na twoim telefonie komórkowym

// Czego potrzebujesz:

telefonu komórkowego,
tabletu lub komputera



// Zadanie

1 **Wybierz oprogramowanie.**
- Np. grę.

2 **Popatrz na oprogramowanie z różnych punktów widzenia.**
- Co działa poprawnie?
- Co działa źle?
- Co działa za wolno?
- Co jest dziwne?

3 **Używaj oprogramowania mając w pamięci te punkty widzenia.**
- Zapisz, co znalazłeś.

4 **Podziel się swoimi odkryciami z innymi.**
- Dyskusja!

Rycerze – programiści i testerzy

Kto testuje najwięcej?

- Zazwyczaj programista (twórca oprogramowania) pisze kod (oprogramowanie), a także dużo testuje.
- Są też testerzy, którzy specjalizują się w testowaniu. Im łatwiej jest wykryć problemy.
- Programiści (deweloperzy) i testerzy razem tworzą zespół wytwarzający oprogramowanie. Tak jak w opowieści fantasy jest wielu rycerzy.
- Większe oprogramowanie zawsze jest tworzone przez pełny zespół, a nie tylko jednego programistę.

Zespół wytwarzający oprogramowanie = grupa ludzi, którzy wspólnie budują i testują oprogramowanie. Krótko mówiąc, zespół wytwórczy.

Kiedy przybywa smok, potrzebujesz…

Opowieść

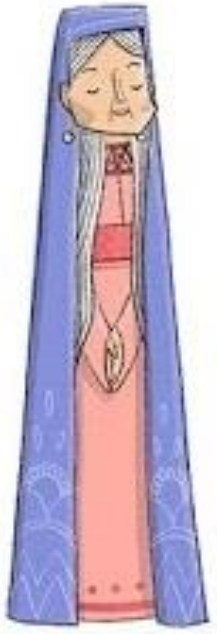
- Laura zawróciła konia i szybko podjechała z powrotem do palisady. Krzyknęła do rycerzy i majstra Adriana, że smok nadchodzi. Wszystkie zaostrzone pnie trzeba było pilnie przenieść do dziury w palisadzie. Włócznie i miecze, ktokolwiek je miał, powinny zostać natychmiast zebrane. Całą dostępną wodę należy wlać do wiader. Potem poszła poszukać Żółtobrodego w zamku.

Objaśnienie

- *W opowieści smok przybywa do wioski w czasie remontu palisady. Podobnie większość defektów znajduje się w oprogramowaniu podczas tworzenia oprogramowania, przed jego wydaniem. Wtedy* ***osoby, które szukają defektów (testerzy) i naprawiają defekty (programiści)*** *są zawsze dostępni. Zwykle tester znajduje defekt, więc nie czeka, aż użytkownik znajdzie defekt później. W tej historii Laura jest testerką, która znalazła i zidentyfikowała defekt, czyli smoka. Jako testerka nie mogła tym razem naprawić usterki, więc potrzebowała pomocy programistów (deweloperów).*

Inni testerzy

Użytkownicy



- Testować może każdy, ale oczekuje się, że programiści (deweloperzy) i testerzy będą testować najwięcej.
- Użytkownicy oprogramowania mogą brać udział w testowaniu, tak jak w opowieściach fantasy mieszkańcy wioski: zarówno dzieci, jak i dorośli, znajdują smoki.
- Ten rodzaj testowania to testy akceptacyjne.
- Użytkownicy mogą również pomóc w tworzeniu oprogramowania.

Tester = osoba, która testuje. Osoba umiejąca testować. Może to być pełnoetatowy tester, programista, administrator lub użytkownik.

Zespół serwisowy

- Zespół serwisowy monitoruje oprogramowanie i utrzymuje je w działaniu.
- Aby to zrobić, zespół serwisowy zarówno testuje, jak i naprawia defekty.
- Pomaga również użytkownikom.
- Zespół serwisowy często próbuje radzić sobie sam, ale w razie potrzeby prosi o pomoc deweloperów, podobnie jak w opowieści fantasy łowca może poprosić rycerzy, aby pomogli mu zabić smoka.
- Czasami osoba zajmująca się konserwacją jest w zespole wytwórczym (zespół wytwórczy staje się zespołem DevOps).

Zespół serwisowy = administratorzy; ci, którzy dbają o działanie oprogramowania, gdy użytkownicy z niego korzystają. Nazywany jest również zespołem operacyjnym.

DevOps – ciągłe dostarczanie

- Czasami w zespole wytwórczym jest osoba operacyjna. W tym przypadku powstaje zespół DevOps, podobnie jak w opowieści fantasy łowcy czasami współpracują z rycerzami.
- Taki zespół cały czas rozwija i testuje oprogramowanie, dostarcza użytkownikom nowe funkcjonalności i jednocześnie wspiera użytkowników oprogramowania w korzystaniu z oprogramowania.

DevOps = łączenie rozwoju oprogramowania i działania oprogramowania. Ten sam zespół buduje i utrzymuje oprogramowanie.

Właściciel produktu

- Właściciele produktów zamawiają oprogramowanie i systemy od zespołów wytwórczych.
- Mogą to być właściciele konkretnych produktów lub zarząd firmy.
- Określają oni, co oprogramowanie powinno robić, ale słuchają też zespołu programistów, podobnie jak w opowieści fantasy Lordowie i Damy rozkazują rycerzom w zamkach i planują razem z nimi.

Właściciel produktu = osoba, która prosi o zbudowanie oprogramowania lub systemu. Właścicielem produktu może być również firma, którą oczywiście reprezentuje jakaś osoba.

Pomoc ekspertów

- Zespoły wytwórcze nie wiedzą wszystkiego, więc potrzebują wsparcia ekspertów w określonych dziedzinach, tak jak w opowieści fantasy mędrcy pomagają wieśniakom i rycerzom.
- Typowi eksperci to specjaliści od użyteczności, bezpieczeństwa i wydajności. Pomagają zespołowi wytwórczemu.
- Na przykład ekspert ds. użyteczności często przeprowadza testy użyteczności lub kieruje nimi.

Ekspert ds. użyteczności = osoba, która specjalizuje się w projektowaniu systemów informatycznych z dobrą użytecznością.

Ćwiczenie 5-15 minut:

Zaprojektuj własnego testera – narysuj własnego rycerza

// Czego potrzebujesz:

papieru i ołówka

// Zadania

1 **Zastanów się, jaki tester byłby dobrym testerem.**
- Ciekawy? Przystojny? Szybki? Cierpliwy?
- Zapisz te cechy.

2 **Pomyśl o odpowiednim rycerzu lub innej postaci, która potrafi znaleźć smoki.**
- Zapisz cechy rycerza.
- W zbroi? Jakie uzbrojenie? Uważny?

3 **Narysuj rycerza.**
- Najważniejsze jest to, abyś zrealizował swój pomysł na to, jak rycerz będzie obrazował testera.
- Nie musisz dążyć do idealnego obrazka.

4 **Pokaż obrazek innym – tak, jak uzgodniliście.**
- Użyj np. Padletu.

Różne rodzaje defektów

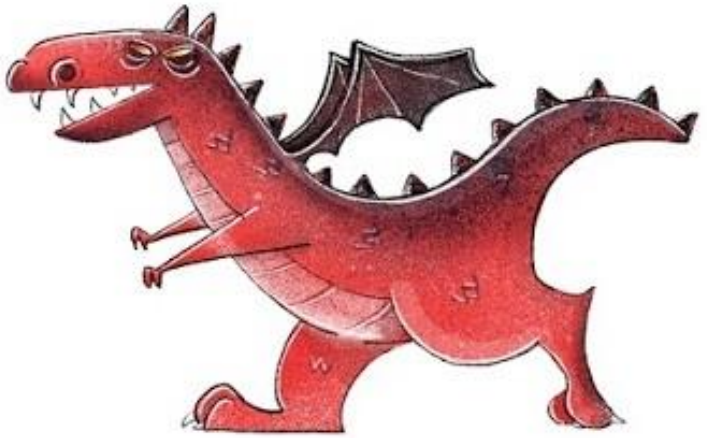
Istnieją różne rodzaje defektów

- Nie możesz znaleźć wszystkich defektów, ale powinieneś spróbować je znaleźć.
- Niektóre defekty są poważne, niektóre małe, niektóre można łatwo usunąć, niektóre są trudniejsze do usunięcia, tak jak w opowieściach fantasy występują różne rodzaje smoków.
- Poważny defekt musi zostać szybko naprawiony.

Ważność = często klasyfikujesz defekty według stopnia ich ważności. Defekt może być poważny, co oznacza, że jego usunięcie jest bardzo kosztowne lub jest bardzo szkodliwy. Defekt może być również mniej poważny.

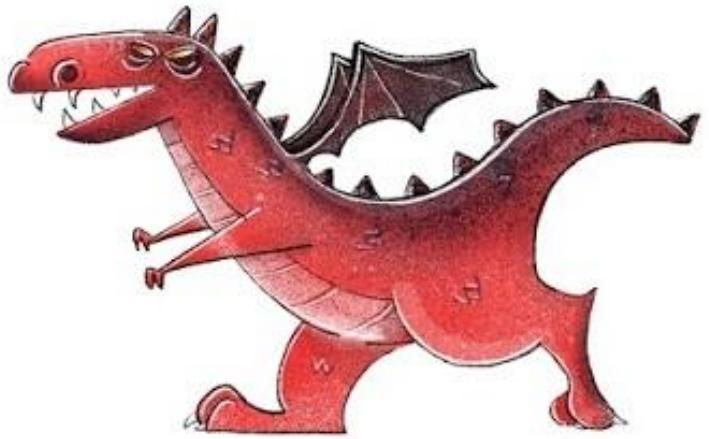
Irytujący czerwony smok

- **Kolor:** czerwony
- **Wielkość:** średni
- **Czy trudny do wytropienia?** Trudny
- **Czy trudny do pokonania?** Łatwy
- **Czy lata?** Nie
- **Skrzydła:** małe
- **Czy zieje ogniem?** Tak
- **Ulubione zajęcia:** jedzenie jagniąt



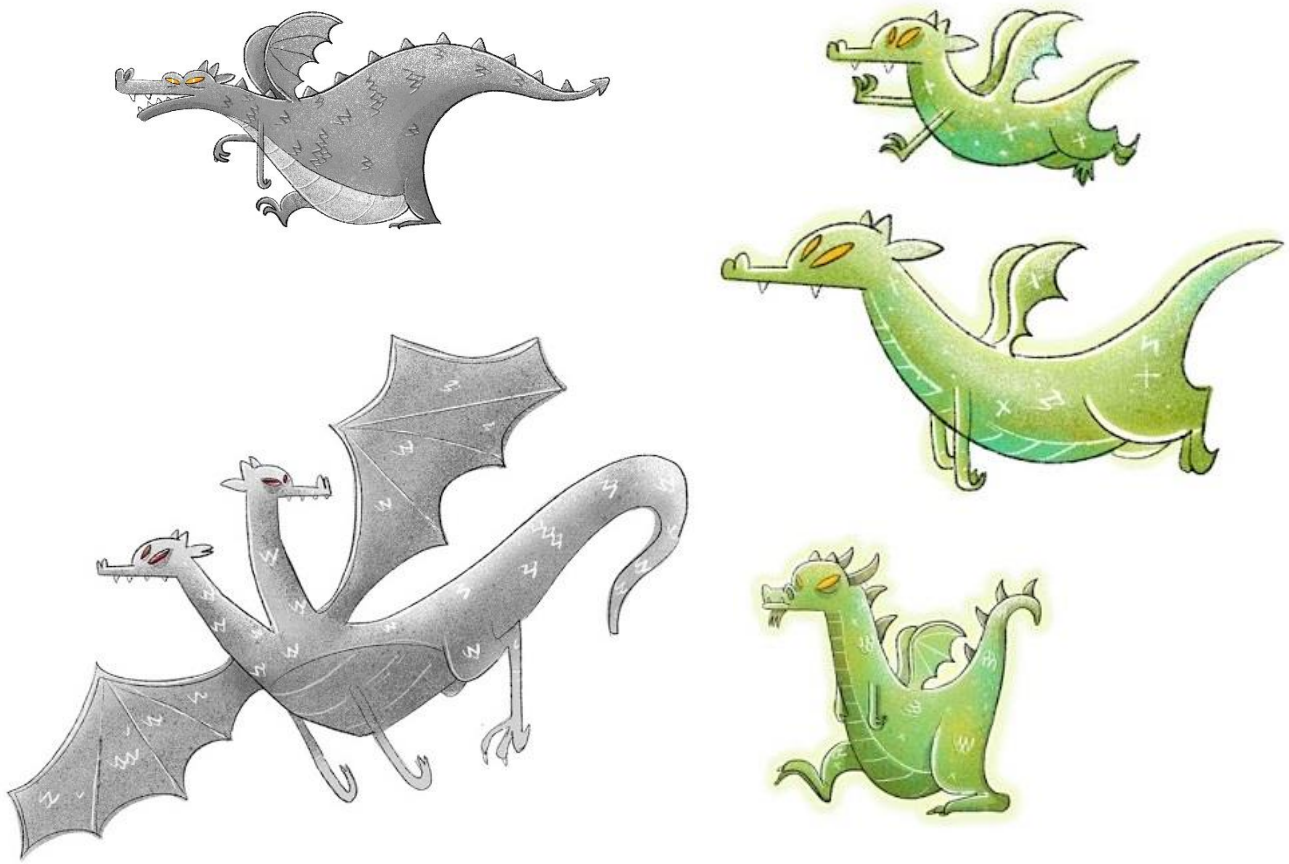
Irytujący czerwony smok

- **Nazwa defektu**: wyciek pamięci
- **Ważność**: średnia
- **Objawy defektu**: komputer działa coraz wolniej, a w końcu przestaje funkcjonować i wyłącza się.
- **Przyczyna defektu**: pamięć jest zarezerwowana na użytek oprogramowania, ale nie jest zwalniana po użyciu.
- **Przyczyna podstawowa**: programista nie zwalnia pamięci we właściwy sposób. Być może nie wie, jak to zrobić, a być może po prostu o tym zapomniał.
- **Testowanie**: mierzysz wykorzystanie pamięci podczas używania oprogramowania. Jeśli ta ilość ciągle rośnie, zapewne w kodzie występuje wyciek pamięci.
- **Usunięcie**: uruchamiasz oprogramowanie linia po linii do momentu znalezienia miejsca, w którym należy wprowadzić poprawkę. Zwalniasz pamięć za pomocą odpowiedniego fragmentu kodu.



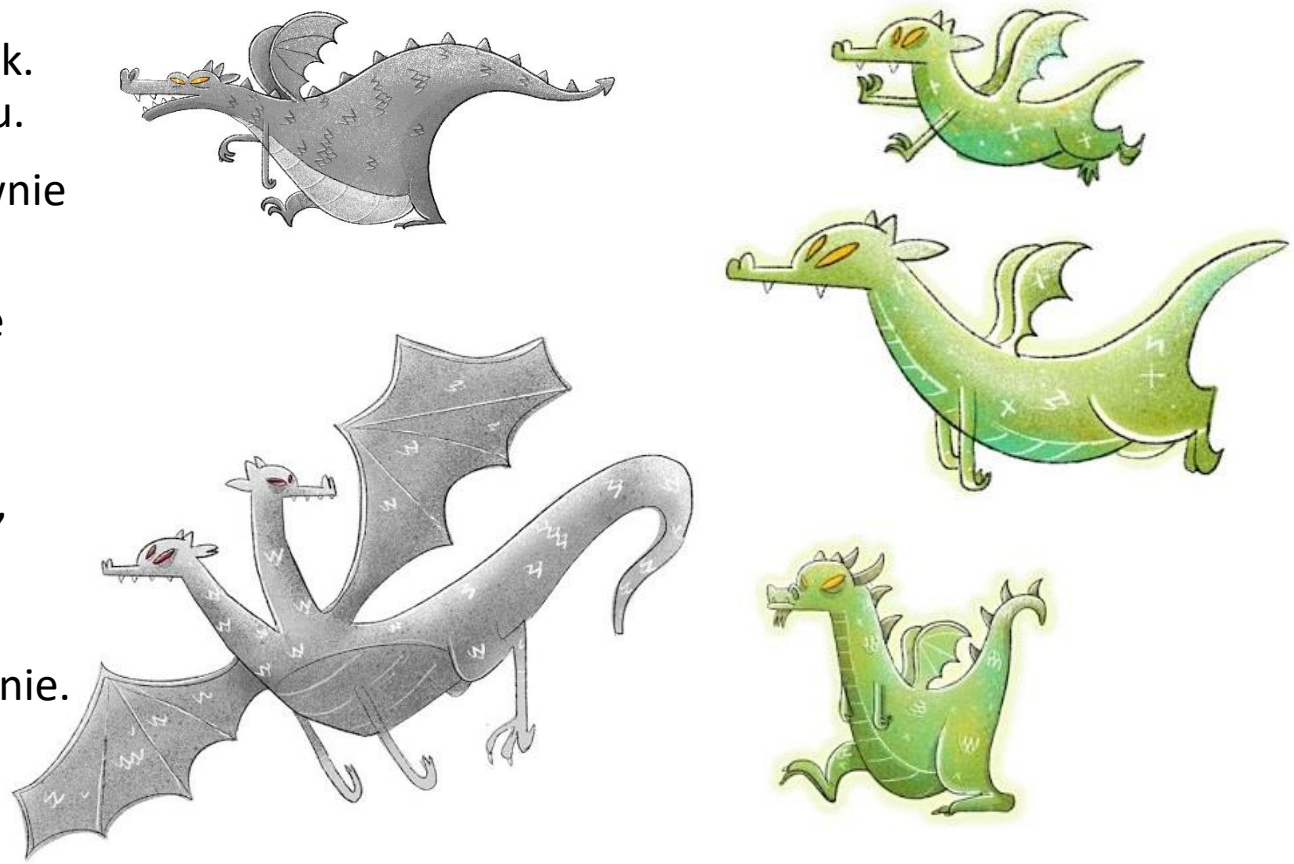
Szary smok łupieżca

- **Kolor:** błyszczący zielony lub szary
- **Wielkość:** mały do dużego
- **Czy trudny do wytropienia?** Raz łatwy, raz trudny
- **Czy trudny do pokonania?** Raz łatwy, raz trudny
- **Czy lata?** Czasami lata, czasami nie
- **Skrzydła:** małe do dużych
- **Czy zieje ogniem?** Tak
- **Ulubione zajęcia:** rabowanie jedzenia i skarbów



Szary smok łupieżca

- **Nazwa defektu**: defekt funkcjonalności
- **Ważność**: mała – średnia - duża
- **Objawy defektu**: oprogramowanie nie robi tego, co powinno. Obliczenie daje niepoprawny (błędny) wynik. Użytkownik widzi informacje w niewłaściwym miejscu.
- **Przyczyna defektu**: funkcjonalność została niepoprawnie (błędnie) zakodowana.
- **Podstawowa przyczyna**: programista (deweloper) nie zrozumiał wymagań użytkownika. Błąd może również wynikać z niedbalstwa programisty albo z pośpiechu.
- **Testowanie**: korzystasz z oprogramowania normalnie, w oparciu o doświadczenie testera lub zdefiniowane wymagania.
- **Usunięcie**: kod zostaje zmieniony, aby działał poprawnie.



Złośliwy czarny smok

- **Kolor:** czarny
- **Wielkość:** mały
- **Czy trudny do wytropienia?** Trudny
- **Czy trudny do pokonania?** Średnio trudny
- **Czy lata?** Tak
- **Skrzydła:** średnie
- **Czy zieje ogniem?** Bardzo!
- **Ulubione zajęcia:** rabowanie w tajemnicy żywności i skarbów

Złośliwy czarny smok

- **Nazwa defektu**: defekt zabezpieczeń
- **Ważność**: duża
- **Objawy defektu**: informacje z oprogramowania znajdują się poza systemem (np. informacje o karcie bankowej). Może to być również nieprawidłowe działanie oprogramowania.
- **Przyczyna defektu**: cyberprzestępca (haker) skorzystał z defektu związanego z zabezpieczeniami, aby włamać się do systemu i coś ukraść albo zniszczyć.
- **Podstawowa przyczyna**: programista nie zastosował najnowszych zaleceń dotyczących tworzenia bezpiecznego kodu. Być może nie wie, jak to zrobić.
- **Testowanie**: wyszukujesz znane luki w oprogramowaniu używając go lub za pomocą oprogramowania do testowania zabezpieczeń. Możesz także przejrzeć kod. Pomocna jest lista kontrolna znanych defektów.
- **Usunięcie**: w przypadku znanego defektu istnieje odpowiednia poprawka. Należy ją zastosować do kodu lub ustawień systemowych.

Szybki purpurowy smok

- **Kolor:** purpurowy
- **Wielkość:** mały
- **Czy trudny do wytropienia?** Trudny
- **Czy trudny do pokonania?** Trudny
- **Czy lata?** Tak
- **Skrzydła:** duże
- **Czy zieje ogniem?** Trochę
- **Ulubione zajęcia:** kradzież złota

Szybki purpurowy smok

- **Nazwa defektu**: defekt wydajności
- **Ważność**: średnia
- **Objawy defektu**: oprogramowanie działa wolniej niż powinno
- **Przyczyna defektu**: część kodu działa nieefektywnie lub po prostu niepoprawnie. Ustawienia też mogą być błędne. Oprogramowanie próbuje wtedy zrobić coś niepotrzebnego. To zabiera czas.
- **Podstawowa przyczyna**: wydajność nie była brana pod uwagę podczas kodowania. Programista może nie znać wszystkich możliwości środowiska programistycznego. Może nie mieć dostępu do pozostałej części oprogramowania.
- **Testowanie**: używasz oprogramowania normalnie, z jednym lub wieloma użytkownikami naraz, często wraz z oprogramowaniem do testowania wydajności. Mierzysz prędkość czyli czas reakcji (odpowiedzi).
- **Usunięcie**: zmieniasz te części kodu, które działają powoli. Próbujesz naprawić i ponownie testujesz prędkość.

Uciążliwy zielony smok

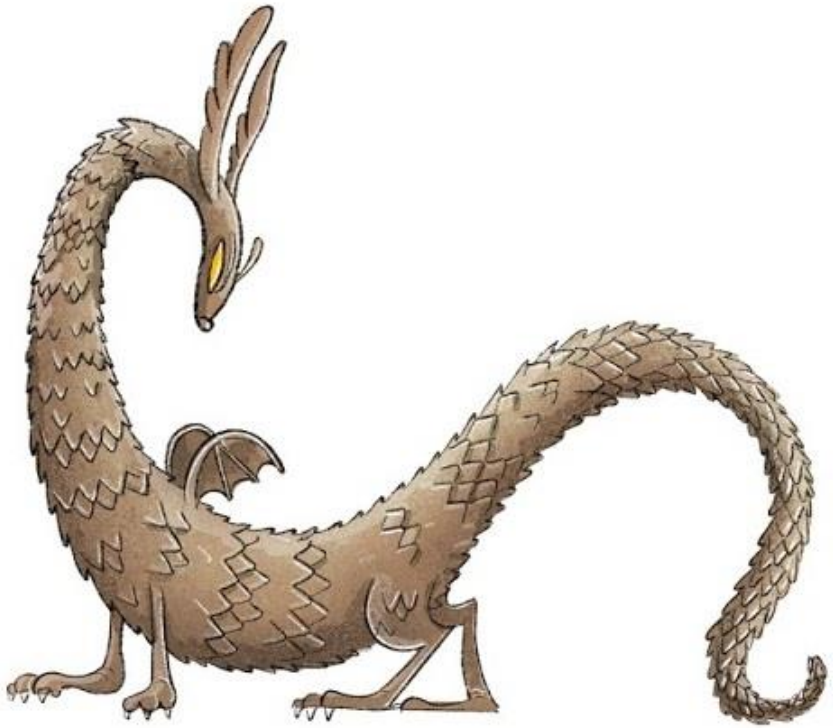
- **Kolor:** zielony
- **Wielkość:** duży
- **Czy trudny do wytropienia?** Łatwy
- **Czy trudny do pokonania?** Trudny
- **Czy lata?** Tak
- **Skrzydła:** duże
- **Czy zieje ogniem?** Bardzo!
- **Ulubione zajęcia:** poszukiwanie złota i pilnowanie terytorium

Uciążliwy zielony smok

- **Nazwa defektu**: defekt użyteczności
- **Ważność**: niska
- **Objawy defektu**: oprogramowanie jest trudne w obsłudze, ale można je użytkować.
- **Przyczyna defektu**: kodowanie zostało wykonane wyłącznie z myślą o funkcjonalności, w możliwie najprostszy sposób.
- **Podstawowa przyczyna**: potrzeby użytkownika nie zostały uwzględnione w projektowaniu lub kodowaniu. Być może użyteczność nie została dobrze zrozumiana.
- **Testowanie**: używasz systemu normalnie. Zbierasz opinie użytkowników o tym, co jest łatwe, a co trudne w użyciu.
- **Usunięcie**: zmieniasz kod, aby system był łatwiejszy w użyciu. Bierzesz przy tym pod uwagę oczekiwania dotyczące testowania użyteczności.

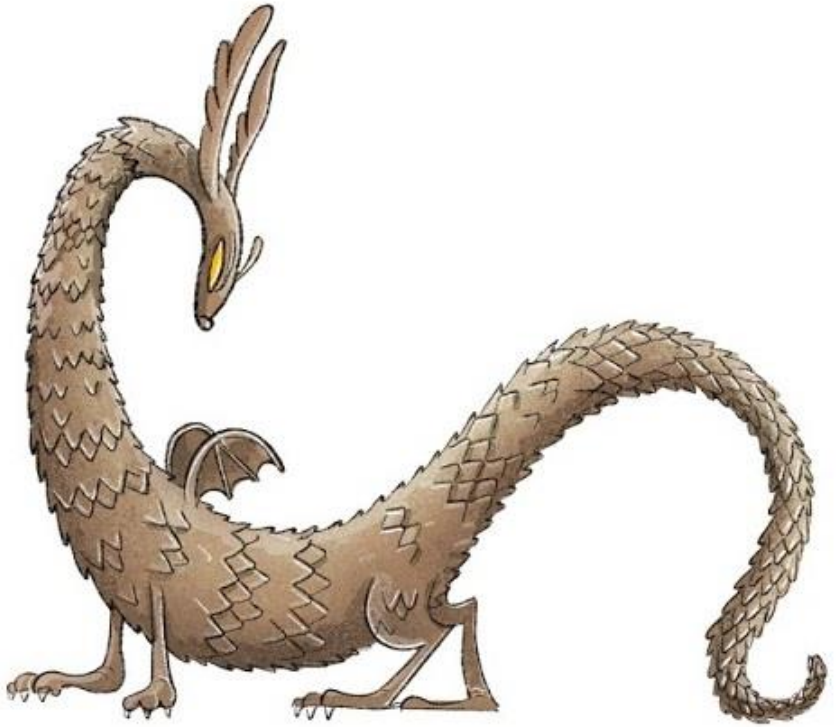
Brązowy smok podziemny

- **Kolor:** brązowy
- **Wielkość:** duży
- **Czy trudny do wytropienia?** Łatwy
- **Czy trudny do pokonania?** Średnio trudny
- **Czy lata?** Nie
- **Skrzydła:** małe
- **Czy zieje ogniem?** Bardzo!
- **Ulubione zajęcia:** poszukiwanie łatwo dostępnego pożywienia i jego pożeranie



Brązowy smok podziemny

- **Nazwa defektu**: defekt sprzętowy
- **Ważność**: wysoka
- **Objawy defektu**: część lub cały komputer nie działa.
- **Przyczyna defektu**: z czasem część sprzętu uległa uszkodzeniu.
- **Podstawowa przyczyna**: część sprzętu może być niskiej jakości, więc nie działa tak długo, jak powinna. Możliwe, że jakaś część nie działa dobrze w zestawieniu z innymi częściami, więc psuje się.
- **Testowanie**: Używasz systemu normalnie. Obserwujesz sprzęt. W środowisku testowym powinno się używać podobnego sprzętu, co użytkownicy.
- **Usunięcie**: Wymieniasz uszkodzoną część na nową lub wymieniasz na część, która lepiej pasuje do pozostałych części.



Znikający szary smok?

- **Kolor:** szary
- **Wielkość:** mały
- **Czy trudny do wytropienia?** Łatwy
- **Czy trudny do pokonania?** Łatwy
- **Czy lata?** Nie
- **Skrzydła:** małe
- **Czy zieje ogniem?** Tak
- **Ulubione zajęcia:** oszukiwanie ludzi

Znikający szary smok?

- **Nazwa defektu**: defekt związany z testowaniem
- **Ważność**: niska
- **Objawy defektu**: wydaje się, że funkcjonalność działa niepoprawnie, np. otrzymujemy błędny wynik obliczeń.
- **Przyczyna defektu**: tester może mieć nieprawidłowe dane testowe lub środowisko testowe.
- **Podstawowa przyczyna**: tester być może zbytnio skupia się na poszukiwaniu defektów, nie przygotował jednak w odpowiedni sposób danych testowych i środowiska testowego.
- **Testowanie**: korzystasz z systemu normalnie, ale obserwujesz środowisko i dane. Zawsze sprawdzaj, czy to testowanie nie jest przyczyną defektu.
- **Usunięcie**: lepiej definiujesz środowisko testowe i dane. Uczysz się na podstawie fałszywych alarmów.

Łagodny zielony smok

- **Kolor:** błyszczący zielony
- **Wielkość:** średni
- **Czy trudny do wytropienia?** Łatwy
- **Czy trudny do pokonania?** Łatwy
- **Czy lata?** Tak
- **Skrzydła:** średnie
- **Czy zieje ogniem?** Tak
- **Ulubione zajęcia:** jedzenie mięsa i pomaganie ludziom

Łagodny zielony smok

- **Nazwa defektu**: posiew usterek, testowanie mutacji - defekt stworzony celowo
- **Ważność**: niska
- **Objawy defektu**: wygląda na to, że funkcjonalność działa nieprawidłowo, np. otrzymujemy błędny wynik obliczeń. Wygląda to jak defekt funkcjonalności.
- **Przyczyna defektu**: tester lub programista celowo utworzył defekt w kodzie.
- **Podstawowa przyczyna**: pomysł polega na tym, że jeżeli znajdziemy wszystkie posiane defekty, to wszystkie defekty powinny zostać znalezione.
- **Testowanie**: używasz systemu normalnie i próbujesz znaleźć wszystkie posiane defekty. Znajdziesz również prawdziwe defekty. Po znalezieniu stworzonego celowo ostatniego defektu można przerwać testowanie.
- **Usunięcie**: pamiętaj, aby naprawić kod również dla posianych defektów, w taki sam sposób jak dla prawdziwych defektów funkcjonalności.

Uparty zielony smok

- **Kolor:** błyszczący zielony
- **Wielkość:** duży
- **Czy trudny do wytropienia?** Łatwy
- **Czy trudny do pokonania?** Trudny
- **Czy lata?** Tak
- **Skrzydła:** małe
- **Czy zieje ogniem?** Tak
- **Ulubione zajęcia:** ciągłe nękanie ludzi

Uparty zielony smok

- **Nazwa defektu**: defekt związany z kontrolą wersji
- **Ważność**: średnia
- **Objawy defektu**: defekt, który został już naprawiony, pojawia się ponownie. Może to być defekt funkcjonalności.
- **Przyczyna defektu**: zarządzanie wersją nie powiodło się. Programista użył starszej części oprogramowania, a należało użyć nowszej części. Ta nowsza część zawiera poprawkę defektu zrobioną przez innego programistę.
- **Podstawowa przyczyna**: pośpiech i zbyt duża liczba ludzi powoduje powstanie defektów. Być może całkowicie brakuje zarządzania wersją.
- **Testowanie**: używasz systemu normalnie. Jeśli znajdujesz defekt, sprawdzasz, czy został już wcześniej usunięty przez programistę. Jeśli tak, sprawdzasz , w jaki sposób usprawnić mechanizmy kontroli wersji.
- **Usunięcie**: Usprawniamy mechanizm kontroli wersji w projekcie. Zacznij korzystać z oprogramowania do kontroli wersji.

Skołowany zielony smok

- **Kolor:** błyszczący zielony
- **Wielkość:** mały
- **Czy trudny do wytropienia?** Łatwy
- **Czy trudny do pokonania?** Łatwy
- **Czy lata?** Tak
- **Skrzydła:** średnie
- **Czy zieje ogniem?** Tak
- **Ulubione zajęcia:** dokuczanie dużym smokom

Skołowany zielony smok

- **Nazwa defektu**: defekt dokumentacji
- **Ważność**: niska
- **Objawy defektu**: funkcjonalność działa inaczej niż zapisano w dokumentacji.
- **Przyczyna defektu**: plany i wytyczne nie są zgodne z oprogramowaniem.
- **Podstawowa przyczyna**: podczas kodowania podjęto decyzje lub wprowadzono zmiany, które spowodowały powstanie innego oprogramowania niż planowano. Dokumentacja została stworzona na podstawie starych planów.
- **Testowanie**: używasz systemu normalnie. Jeśli zostanie znaleziona usterka, ustalamy, czy niepoprawne jest oprogramowanie, czy dokumentacja.
- **Usunięcie**: zmieniasz oprogramowanie lub dokumentację, aby były ze sobą zgodne.

Ćwiczenie 5-15minut:

zaprojektuj własny dobry defekt–
narysuj własnego dobrego smoka

// Czego potrzebujesz:

papieru i ołówka

// Zadania

1 **Pomyśl o defekcie, który pomógł ci zrozumieć jakiś aspekt działania oprogramowania (lub możesz wymyślić jakikolwiek defekt, jeśli chcesz).**

- Zapisz nazwę defektu i opisz go w kilku słowach.

2 **Pomyśl o odpowiednim smoku.**

- Zapisz cechy smoka.
- Jeśli defekt był poważny, smok jest duży itp.

3 **Narysuj smoka.**

- Najważniejsze jest to, abyś zrealizował swój pomysł na to, jak smok będzie reprezentował defekt.
- Nie musisz dążyć do idealnego obrazka.

4 **Pokaż obrazek innym – tak, jak uzgodniliście.**

- Użyj np. Padletu.

Dziękujemy!

Jeżeli interesuje Cię książka, możesz zamówić ją tutaj (wersja **polska**):

https://ksiegarnia.pwn.pl/Smoki-fora-ze-dwora,961185040,p.html

Śledź i udostępniaj informacje o projekcie:

https://www.dragonsout.com

Masz pytania:

kontakt@sjsi.org

lub

kari.kakkonen@dragonsout.com

Jeżeli interesuje Cię książka w wersji angielskiej, możesz zamówić ją tutaj:

https://www.dragonsout.com/p/order-dragons-out-book.html

Śledź i udostępniaj informacje o projekcie:

https://www.dragonsout.com

Masz pytania:

kari.kakkonen@dragonsout.com

lub

kontakt@sjsi.org