Magazine

Podejścia anty-regresyjne: Analiza wpływu i porównawcze oraz mieszane testowanie regresyjne

Część II: Zapobieganie i wykrywanie regresji przy użyciu technik statycznych

Autor: Paul Gerrard

O autorze: Paul jest konsultantem, nauczycielem, autorem, projektantem stron, programistą, testerem, prelegentem na konferencjach, coachem i wydawcą. Podejmował się usług konsultingowych we wszystkich aspektach testowania oprogramowania i zapewnienia jakości, specjalizując się w zapewnieniu testów. Prezentował wykłady i przewodniki na konferencjach w całej Europie, USA, Australii i Południowej Afryce i od czasu do czasu zdobywał za nie nagrody.

Wykształcony na uniwersytetach w Oxford i Imperial College London, Paul był założycielem Schematu Certyfikacji Testerów BCS ISEB oraz członkiem Grupy roboczej opracowującej BS 7925 – Standard dla testowania komponentów. Obecnie jest dyrektorem Gerrard Consulting Limited, dyrektorem Aqastra Limited i gospodarzem Forum Zarządzania Testowaniem w Wielkiej Brytanii.

Intermediate

Level

4

Magazine Number

Testowanie oprogramowania

Section in the magazine

Wprowadzenie

W pierwszej części niniejszego cyklu artykułów, przyjrzeliśmy się naturze regresji i analizie wpływu. W bieżącym artykule przedstawimy analizę wpływu jako technikę zapobiegania regresji; w większym stopniu porównany analizę wpływu technicznego i biznesowego, a także omówimy zapobieganie i wykrywanie regresji przy użyciu analizy wpływu biznesowego oraz statycznej analizy kodu. Kolejny artykuł (Część III) skupi się na testowaniu regresyjnym jako podejściu wykrywania regresji.

Zapobieganie i wykrywanie regresji

Zanim posuniemy się dalej, warto zbadać powiązanie pomiędzy analizą wpływu (używaną do zapobiegania regresji) a testowaniem (używanym do wykrywania regresji). Przyglądaliśmy się analizie wpływu zarówno z biznesowego, jak i technicznego punktu widzenia, ale możemy również porównać czynności „przed zmianą" w analizie wpływu z czynnościami „po zmianie" w testowaniu.

	Techniczny punkt widzenia (Oparty na projekcie lub kodzie)	**Biznesowy punkt widzenia (Oparty na zachowaniu)**
Analiza wpływu "przed zmianą" (zapobieganie regresji)	**Analiza wpływu technicznego** Manualna analiza projektów i kodu źródłowego celem określenia potencjalnego wpływu zmiany.	**Analiza wpływu biznesowego** Domniemanie, oparte na obecnym zachowaniu system I kontekstu biznesowego, potencjalnego wpływu zmiany.
Testowanie "po zmianie" (wykrywanie regresji)	**Statyczne testowanie regresyjne** Automatyczna analiza statyczna kodu źródłowego celem identyfikacji odchyleń od analizy po zmianie.	**Dynamiczne testowanie regresyjne** Wykonanie przygotowanych testów dynamicznych celem porównania nowego zachowania z poprzednim.

Tabela podsumowuje w macierzy 2x2 cztery antyregresyjne czynności. Pomiędzy analizą wpływu biznesowego i technicznego istnieją pewne podobieństwa. Obie opierają się na obecnym rozumieniu istniejącego systemu (ale na poziomie technicznym lub biznesowym zależnym od punktu widzenia). Obie są w pewnym sensie domniemaniem i skupiają się na tym, jak można uniknąć czy obsłużyć regresję w danej technologii czy procesie biznesowym.

Techniki „po zmianie" skupiają się na tym, jak można wykryć regresje. Opierają się na dowodach wyodrębnionych z analizy „przed zmianą" projektu i kodu lub demonstracji zaimplementowanej funkcjonalności przy użyciu uprzednio uruchomionego zestawu testów.

Kompletna strategia antyregresyjna powinna zawierać wszystkie z tych czterech technik.

Analiza wpływu technicznego (zapobieganie regresji)

Analiza wpływu technicznego jest w skrócie przeglądem przewidywanej zmiany w projekcie systemu na wysokim poziomie i może przyjąć formę przeglądu technicznego. Tam, gdzie rozważane są poważne rozszerzenia, a zmiany, które mają być wykonane, są główne dodatkami do istniejącego systemu, przegląd techniczny skupi się na wpływie na poziomie architektonicznym (ryzyka dotyczące wydajności, bezpieczeństwa, odporności). Tego nie będziemy dalej omawiać.

Na poziomie kodu rozważania powinny skupić się na nowych interfejsach (bezpośrednia integracja) i zmianach w zasobach współdzielonych, takich jak bazy danych (pośrednia integracja). Oczywiście, zmiany, które mają być wykonane w istniejącej bazie kodu, jeśli są znane, muszą być dość szczegółowo przestudiowane.

Projektanci i programiści muszą wykonać pewną formę analizy kodu źródłowego na wersji oprogramowania sprzed zmian. Analiza ta jest po prostu inspekcją lub przeglądem kodu. Programista spekuluje na temat tego, jaki mógłby być wpływ zmian i śledzi ścieżki wykonania w kodzie i innych niezmienionych modułach, aby sprawdzić, jaki wpływ mogą mieć dane zmiany. Zmiany projektowe obejmujące schemat bazy danych, formaty wiadomości, mechanizmy wywołań etc. mogą wymagać zmian w wielu miejscach. Mogą pomóc proste narzędzia wyszukiwania i skanowania, lecz jest to oczywiście czynnością pracochłonną i podatną na błędy.

W małych próbkach kodu, które są proste, dobrze zaprojektowane i gdzie jest mało modułów, z którymi współpracują, programista będzie miał realną szansę zidentyfikowania potencjalnych problemów, które leżą blisko obszaru proponowanych zmian. Wykryte anomalie mogą być wyeliminowane poprzez dostosowanie projektu zmian, które mają być wykonane (lub poprzez przyjęcie projektu, który unika ryzykownych zmian). Jednakże w systemach o realistycznych rozmiarach, skala i złożoność tego zadania drastycznie ogranicza jego zasięg i efektywność.

Pewne zmiany będą wykryte przez kompilatory lub procesy buildowe używane przez programistów, te są mniejszym zmartwieniem. Wpływ bardziej subtelnych zmian może wymagać pewnej „pracy detektywistycznej”. Zwykle programista może być zmuszony do wyszukiwania w całej bazie kodu, by znaleźć specyficzne wzorce kodu. Typowe przykłady to dostęp do tabeli w zmienionej bazie danych; użycie nowego lub zmienionego elementu w sformatowanym komunikacie XML; użycie zmiennej, która ma zmieniony zakres walidacji itd.

Zwykle formalna analiza statyczna kodu źródłowego programu wykonywana jest przy użyciu narzędzi. Nie może to być użyte do przewidywania wpływu na zachowanie zmian (dopóki zmieniony kod jest w trakcie analizy, a przyjrzymy się temu w dalszej części), ale wynik uzyskiwany z narzędzi może być wykorzystany do skupienia uwagi programisty na specyficznych aspektach systemu. Nie trzeba mówić, że nieoceniona jest baza defektów używana do identyfikacji modułów podatnych na błędy w repozytorium kodu. Na obszary wymagające specjalnej uwagi można przeznaczyć więcej wysiłku w inspekcji potencjalnych efektów ubocznych zmian.

Analiza wpływu biznesowego (zapobieganie regresji)

Kiedy do nowej systemu należy dodać dodatkową funkcjonalność lub wymagane jest rozszerzenie do istniejącej funkcjonalności, wtedy trzeba wykonać pewien rodzaj analizy wpływu biznesowego, kierowanej przez zrozumienie proponowanego i istniejącego zachowania. Sporadycznie naprawa błędu wymaga znacznych zmian projektowych, ma więc miejsce przegląd obszarów funkcjonalnych, które mają być zmienione i obszarów funkcjonalnych, które mogą być dotknięte zmianą.

Analiza wpływu biznesowego jest tak naprawdę zestawem pytań “co-jeśli” zadanych w odniesieniu do istniejącego systemu przed wykonaniem proponowanych zmian. Odpowiedzi do tych pytań mogą spowodować wątpliwości co do trybów potencjalnych awarii – konsekwencji – w stosunku do których powinna być wykonana analiza ryzyka. Oczywiście – liczba potencjalnych trybów awarii jest wielka, a wiedza potrzebna do przeanalizowania tych ryzyk może być bardzo ograniczona. Jednakże obszary największego zainteresowania mogłyby być przedstawione programistom, aby ci zwrócili na nie specjalną uwagę i w szczególny sposób skupili się na testowaniu regresyjnym.

Analiza wpływu biznesowego przebiega według dość spójnego procesu:

PROPOZYCJA: Po pierwsze, proponowane rozszerzenie, zmiana lub rozwiązanie błędu jest opisywane i komunikowane użytkownikom biznesowym.

KONSEKWENCJE: Następnie użytkownicy biznesowi rozważają zmiany w funkcjonalności oraz zmiany techniczne, o których programiści wiedzą, że będą miały wpływ na określone aspekty funkcjonalne systemu. Czy istnieją potencjalne tryby awarii wymagające szczególnego zainteresowania?

WYZWANIE: Wreszcie, użytkownicy biznesowi stawiają wyzwania programistom i pytają "co by sie stało, gdyby…" aby wyjaśnić niepewności i przedstawić wszelkie specyficzne testy regresji, które powinny być odpowiednie do rozwiania ich obaw.

Głównym wynikiem tego procesu może być zbiór deklaracji, które skupią uwagę programistów. Często specyficzne cechy lub procesy mogą być uważane za krytyczne i w żadnych okolicznościach nie mogą być niekorzystnie dotknięte przez zmiany. Takie cechy i procesy z pewnością będą przedmiotem późniejszego testowania regresyjnego.

Statyczne testowanie regresyjne (wykrywanie regresji)

Czy narzędzia do analizy statycznej mogą pomóc w działaniu antyregresyjnym? Niniejsza sekcja sugeruje, jak można użyć narzędzi do analizy statycznej, by wyróżnić zmiany, które mogą być źródłem obaw. Propozycja ta jest raczej spekulacją – bylibyśmy bardzo zainteresowani poznaniem opinii praktyków lub badaczy zajmujących się tym obszarem.

Zazwyczaj narzędzia są używane do skanowania nowego lub zmienionego kodu źródłowego, skupiając się na wykryciu słabych praktyk programistycznych i defektów wykrywalnych statycznie, tak by programiści mogli je wyeliminować. Jednakże regresje są znajdowane często w kodzie niezmienionym, który wywołuje lub jest wywoływany przez kod zmodyfikowany. Skanowanie kodu w niezmienionym module nie powie Ci niczego, czego byś nie wiedział. Tak więc analiza musi zajrzeć w zmieniony kod i śledzić ścieżki dotknięte zmianami w niezmienionym module, które wywołują lub są wywoływane przez zmieniony moduł. Oczywiście mogą istnieć ekstremalnie złożone interakcje, z którymi mają się zmierzyć narzędzia – ale właśnie po to takie narzędzia istnieją.

Proces ten wyglądałby mniej więcej w taki sposób:

Na niezmienionym kodzie wykonana jest analiza całej bazy kodu lub wybranych komponentów systemu, który ma być zmieniony (zakres musi być zdefiniowany i spójny w tym procesie).

Ta sama analiza jest wykonana na zmienionej bazie kodu.

Obydwie analizy są porównywane - wykazane są różnice celem określenia miejsc, w których zmiany kodu wpływają na strukturę i ścieżki wykonywania systemu.

Narzędzie porównujące powie Ci, co zmieniło się w kodzie. Identyfikacja różnic w wynikach analizy statycznej może pomóc Ci zlokalizować miejsca, na które zmiany kodu mają wpływ.

Obecnie narzędzia mogą generować wiele typów analiz. Jakie typy analiz mogą być przedmiotem zainteresowania?

Analiza przepływu sterowania: jeśli dwie analizy przepływu sterowania zmienionego systemu się różnią, ale różnice występują w niezmienionym kodzie, wtedy możliwie jest, że wykonywana jest część kodu, która nie była wcześniej używana, lub też część kodu używana wcześniej nie jest już wykonywana. Nowy przepływ sterowania w oprogramowaniu może być zamierzony, ale rzecz jasna, nie musi tak być. Analiza ta daje po prostu programistom wskaźnik do funkcjonalności i ścieżek w kodzie, które są warte dalszego badania. Jeśli narzędzie może generować graficzne wykresy przepływu sterowania, dla wprawnego oka zmiany mogą być oczywiste. Proces może być analogiczny do lekarza badającego RTG przeprowadzonego w różnym czasie w poszukiwaniu wzrostu nowotworu lub leczenia złamanej kości.

Analiza przepływu danych: analizy przepływu danych śledzą użycie przypisania zmiennych, orzeczeń w decyzjach (np. z odniesieniem do jeśli…wtedy…w przeciwnym razie…stwierdzenie) lub wartości zmiennej użytej w jakiejś innej operacji, takie jak przypisanie do innej zmiennej lub użytej w wyliczeniu. Różnica we wzorcu użycia zmiennej zdefiniowanej w zmienionym module, przekazana z lub od niezmienionych modułów może wskazywać na niepożądane zmiany z zachowaniu oprogramowania.

Niezbyt wiele organizacji używa narzędzi do analizy statycznej i zbyt mało narzędzi jest zaprojektowanych do wyszukiwania różnic w wynikach “analizy głębokości przepływu” pomiędzy wersjami całych systemów – lecz wyraźnie istnieją pewne potencjalne możliwości eksploracji na tym polu. Niniejszy artykuł pozostawia Cię z pewnymi koncepcjami ale nie rozwinie głębiej tych sugestii.

Jak dotąd przyjrzeliśmy sie trzem technikom zapobiegania i wykrywania regresji. W następnym artykule zbadamy czynność bardziej interesującą programistów i testerów – testowanie regresyjne.

Ciąg dalszy nastąpi…