Proponowane tematy prac magisterskich z inżynierii oprogramowania (ang. MSc Theses Proposals) 2009/2010
Data: 2008-12-10
Opiekun: dr inż. Lech Madeyski
E-Mail: lech [< d o t >] m a d e y s k i [< a t >] p w r [< d o t >] w r o c [< d o t>] pl
WWW: http://madeyski.e-informatyka.pl

Zgłaszający temat: dr inż. Lech Madeyski
Temat: Praktyka Programowania Poprzez Ciągłe Testowanie (ang. *Continuous Test-Driven Development Practice*)
Programowanie poprzez testy (Test-Driven Development lub Test-First Programming) jest wypromowaną przez metodykę XP praktyką wytwarzania oprogramowania, która staje się coraz popularniejsza. Praktyka CTDD (Continuous Test-Driven Development) polega na asynchronicznym, ciągłym uruchamianiu i wykonywaniu testów podczas tworzenia kodu i informowaniu programisty o ich rezultacie. Celem pracy jest:

- Przedstawienie przeglądu literatury i istniejących narzędzi wspomagających programistę podczas stosowania praktyki TDD oraz Continuous TDD.
- Stworzenie narzędzia w postaci wtyczki (plugin) do środowiska programistycznego Eclipse wspomagającej programistę w stosowaniu i doskonaleniu praktyki Continuous TDD z uwzględnieniem testów jednostkowych i akceptacyjnych.
- Empiryczna analiza wpływu stosowania praktyki Continuous TDD (z wykorzystaniem stworzonego narzędzia) na produkty i proces wytwarzania oprogramowania w środowisku akademickim i/lub przemysłowym.

Rodzaj pracy: praktyczno-teoretyczna
Wymagania: Wymagana dobra znajomość Javy, JUnit, Eclipse IDE. Znajomość Fit/Fitness, TestNG i umiejętność tworzenia pluginów do Eclipse będą dodatkowymi atutami. Mile widziane doświadczenie w realizacji większych projektów, chęć do pracy oraz bierna znajomość języka angielskiego.
Liczba realizatorów: Praca jest przeznaczona dla jednej osoby.

Zgłaszający temat: dr inż. Lech Madeyski
Temat: Testowanie Mutacyjne (ang. *Mutaton Testing*)
Omówienie: Testowanie jest kluczową metodą wpływania na jakość tworzonego oprogramowania. Testowanie mutacyjne jest zaś bardzo obiecującą metodą badania jakości testów. Celem pracy jest:

- Przedstawienie stanu sztuki i istniejących rozwiązań w dziedzinie testowania mutacyjnego.
- Stworzenie lub rozwinięcie istniejącego narzędzia do testowania mutacyjnego o nowe (wskazane przez promotora) mutacje dla języka Java.
- Podjęcie próby częściowego rozwiązania problemu ekwiwalentnych mutantów.
- Zapewnienie wysokiej wydajności procesu testowania mutacyjnego.
- Empiryczna ocena narzędzia i zaimplementowanych mutacji w środowisku akademickim i/lub przemysłowym.

Rodzaj pracy: praktyczno-teoretyczna
Wymagania: chęć do pracy i zdolność do twórczego działania, wysokie umiejętności programistyczne i dobra znajomość Javy oraz bierna znajomość języka angielskiego. Znajomość AspectJ będzie dodatkowym atutem.
Liczba realizatorów: Praca jest przeznaczona dla jednej osoby.

Zgłaszający temat: dr inż. Lech Madeyski
Temat: Ciągły Przegląd Kodu w Metodykach Zwinnych – Propozycja Nowej Praktyki (ang. *Agile Continuous Code Review – a New Practice Proposal*)
Omówienie: Regularne przeglądy kodu przyczyniają się do poprawy jakości tworzonego oprogramowania. kodu. Mogą również przyczynić się do redukcji kosztów wytwarzania oprogramowania poprzez wczesne wykrywanie błędów. Skrajną formą częstego przeglądu kodu jest ciągły przegląd kodu. W metodyce eXtreme Programming (XP) jest to realizowane za pomocą praktyki programowania parami (pair-programming), czyli dwóch programistów pracujących razem. Stosowanie tej praktyki jest jednak dość kosztowne. Nie wszyscy lubią lub mogą być w jednym miejscu by programować w parach. Drugi programista nie ma wystarczającego dystansu pozwalającego niekiedy dostrzec problemy. Dlatego w niniejszej pracy proponuje się nową praktykę Agile Continuous Code Review (ACCR) polegającą na wykorzystaniu dedykowanego narzędzia wspomagającego przegląd kodu z pomocą zewnętrznego recenzenta (w szczególnie ważnych momentach np. podczas nanoszenia skomplikowanych zmian lub modyfikacji w „stabilnej" gałęzi kodu) jak i narzędzia zapewniającego automatyczną analizę kodu (np. wykorzystując statyczne analizatory kodu, narzędzia zwracające metryki kodu, tester mutacyjny).
Celem pracy jest:

- Przegląd literatury związanej z tematyką pracy.
- Propozycja procesów przeglądu kodu z pomocą zewnętrznego recenzenta i automatycznego narzędzia do przeglądu kodu.
- Implementacja powyższych procesów w postaci wtyczki (plugin) do środowiska programistycznego Eclipse. Wtyczka powinna rozszerzalną architekturę i zapewnić łatwą integrację z popularnymi narzędziami do automatycznej analizy kodu (np. narzędzia do statycznej analizy kodu).
- Empiryczna ocena nowej praktyki i stworzonego narzędzia w środowisku akademickim i/lub przemysłowym.

Rodzaj pracy: praktyczno-teoretyczna
Wymagania: chęć do pracy i zdolność do twórczego działania, wysokie umiejętności programistyczne i dobra znajomość Javy, bierna znajomość języka angielskiego. Znajomość AspectJ będzie dodatkowym atutem.
Liczba realizatorów: Praca jest przeznaczona dla jednej osoby.

Zgłaszający temat: dr inż. Lech Madeyski
Temat: Budowa Teorii w Inżynierii Oprogramowania (ang. *Building Theories in Software Engineering*)
Omówienie: Budowa teorii empirycznych jest stosunkowo młodym nurtem w inżynierii oprogramowania. Technology Acceptance Model (TAM) jest empirycznie zweryfikowaną teorią próbującą zamodelować czynniki wpływające na podjęcie decyzji, przez przyszłych użytkowników, czy będą używać danej technologii lub oprogramowania czy też nie. Oprogramowanie jest tworzone dla użytkowników, dlatego odnosi ono sukces dopiero wtedy gdy zostanie zaakceptowane i będzie używane.
Celem pracy jest:

- Przegląd literatury związanej z tematyką pracy.
- Stworzenie narzędzia (aplikacji internetowej) do przeprowadzania badań empirycznych na potrzeby formułowanych teorii (w szczególności TAM).

- Wykorzystanie narzędzia do przeprowadzania badań empirycznych (np. dotyczących akceptacji narzędzi, technologii, metodyk, praktyk w inżynierii oprogramowania przez przyszłych użytkowników) w środowisku akademickim i/lub przemysłowym.
- Propozycja nowej teorii i próba jej empirycznej oceny w środowisku akademickim i/lub przemysłowym.

Rodzaj pracy: praktyczno-teoretyczna
Wymagania: chęć do pracy i zdolność do twórczego działania, wysokie umiejętności programistyczne i dobra znajomość Javy, bierna znajomość języka angielskiego.
Liczba realizatorów: Praca jest przeznaczona dla jednej osoby.
Prowadzący: dr inż. Lech Madeyski

Zgłaszający temat: dr inż. Lech Madeyski
Temat: Przewidywanie Błędów w Oprogramowaniu na Podstawie Metryk Kodu (ang. *Software Metrics for Predicting Faults*)
Omówienie: Duże firmy wytwarzające oprogramowanie borykają się z ogromnymi kosztami, których przyczyną są błędy wykrywane dopiero podczas użytkowania systemu. Kluczowym zagadnieniem jest możliwość wskazywania tych fragmentów systemu (często klas), które powinny być poddane przeglądowi/inspekcji ze względu na duże prawdopodobieństwo wystąpienia błędów.
W ramach pracy należy:

- Scharakteryzować metody zapewniania jakości tworzonego oprogramowania oraz narzędzia umożliwiające zgłaszanie błędów (ang. *bug trackers*).
- Zaproponować metryki, które mogą służyć do wskazywania klas, które powinny być poddane inspekcji ze względu na duże prawdopodobieństwo wystąpienia błędów.
- Zaimplementować narzędzie współpracujące z systemem (lub systemami) zgłaszania błędów.
- Zaimplementować narzędzie do obliczania metryk kodu lub wykorzystać gotowe narzędzie (pod warunkiem, że zwracane metryki zostaną dokładnie zweryfikowane).
- Empirycznie zbadać efektywność metryk na wybranych projektach w środowisku open source lub przemysłowym.

Rodzaj pracy: praktyczno-teoretyczna
Wymagania: chęć do pracy i zdolność do twórczego działania, wysokie umiejętności programistyczne i dobra znajomość Javy, bierna znajomość języka angielskiego.
Liczba realizatorów: Praca jest przeznaczona dla jednej osoby.

Zgłaszający temat: dr inż. Lech Madeyski
Temat: Refaktoryzacja do Wzorców Projektowych (ang. *Refactoring to Patterns*)
Omówienie: Wzorce projektowe stanowią zbiór uniwersalnych, sprawdzonych i gotowych rozwiązań pewnej kategorii problemów występujących w trakcie wytwarzania oprogramowania. Prawidłowe zastosowanie wzorców projektowych pozwala na stworzenie wyższej jakości i bardziej elastycznego systemu.
W ramach pracy należy:

- Przegląd literatury związanej z tematyką pracy.
- Stworzenie narzędzia (wtyczki) do środowiska programistycznego Eclipse umożliwiającego:
    - Generowanie hierarchii klas, interfejsów oraz metod odpowiadających określonym wzorcom projektowym.

    - Dodanie przez użytkownika szablonu własnego wzorca projektowego, na podstawie którego możliwe będzie generowanie struktury klas oraz interfejsów.
    - Refaktoryzację istniejącego kodu do określonego wzorca projektowego z zachowaniem dotychczas osiągniętej funkcjonalności.
    - Sugerowanie użytkownikowi refaktoryzacji kodu do określonego wzorca projektowego na podstawie analizy powstałego już kodu.
- Empirycznie zbadać efektywność stosowania nowego narzędzia na wybranych projektach w środowisku akademickim, open source lub przemysłowym.

Rodzaj pracy: praktyczno-teoretyczna
Wymagania: chęć do pracy i zdolność do twórczego działania, wysokie umiejętności programistyczne i dobra znajomość Javy, bierna znajomość języka angielskiego.
Liczba realizatorów: Praca jest przeznaczona dla jednej osoby.
Uwaga: Temat dodatkowy (za zgodą prof. Huzara).