Politechnika Wrocławska

Zwinne metodyki wytwarzania oprogramowania

Lech Madeyski
Lech.Madeyski@pwr.wroc.pl
http://madeyski.e-informatyka.pl/
Politechnika Wrocławska
Instytut Informatyki
Zakład Inżynierii Oprogramowania

O mnie

- I-32/ZIO,
- IEEE/ACM/PTI/IR/...
- e-Informatica Software Engineering Journal – co-Editor-in-Chief http://www.e-informatyka.pl/e-Informatica
- SEnS http://www.e-informatyka.pl/sens/
- Konkurs na najlepszą pracę magisterską z informatyki

O mnie

Zainteresowania badawcze:

- Empirical Studies in Software Engineering,
- Software Metrics and Quality Models,
- Software Quality, Software Process Improvement,
- Agile Software Development Methodologies and Practices (e.g. eXtreme Programming, TDD),
- Data mining in Software Engineering,
- Defect Prediction Models,
- Aspect-Oriented Programming,
- Web Applications Architectures, Design Patterns, Technologies.

O przedmiocie

- Tygodniowa liczba godzin zajęć:
  - 1W
  - 2P
  - 1S

Konsultacje

- Wtorek 11-13
- Czwartek 10-11, 17-18

Pokój 415 B-4

Wykład

- 11X i 25X – Szybki start do realizacji projektów zgodnie ze Scrum/XP, czyli workshop organizowany we współpracy Capgemini z kołem naukowym inżynierii oprogramowania SEnS (http://www.e-informatyka.pl/sens/)
- Kolejne wykłady – dalsze zgłębianie tajników agile (testowanie, metodyka i praktyki XP, Scrum, Kanban)

Wykład

- 22XI, 6XII, 20XII: chętni w grupach 1-4 os. mogą zgłaszać 10-45 min. warsztaty, demonstracje, gry, zabawy, prezentacje etc. mające na celu przybliżenie i doskonalenie znajomości praktyk, zasad, wartości, technik, narzędzi stosowanych w dowolnych metodykach zwinnych
- [+0..10 procent punktów z kolokwium]-zapisy mailem (temat, czas i kompletna treść prezentacji).
- Decyduje atrakcyjność propozycji (mogą zostać odrzucone) i kolejność zgłoszeń.

Wykład - zaliczenie i wymagania wstępne

- Ostatnie dwa wykłady (3l i 17l): pierwszy i drugi termin kolokwium
- Wymagania wstępne zaliczenia wykładu:
  - Student **nie** musi zaliczyć zajęć towarzyszących, ale musi uczestniczyć w wykładach i stawić się na kolokwium.
- Zasady dyskwalifikacji:
  - Student otrzymuje ocenę niedostateczną, jeżeli w czasie kolokwium zostanie przyłapany na korzystaniu ze ściąg lub pomocy kolegów.

Projekt (zespołowy 4 os.) - cel 1:

Propozycja, stworzenie i ewaluacja **nowego** narzędzia (np. w powiązaniu z nową lub istniejącą praktyką) wspomagającego pracę członków zespołów projektowych używających zwinnych metodyk i praktyk.

Waga oceny: 50% punktów

Kiedy ocena? Propozycja rozwiązania i jego oceny (2,3 zajęcia) [2x0..5 pkt] + finalna ocena na przedostatnich (14) zajęciach [0..30 pkt] + badanie empiryczne [0..10 pkt].

Co podlega ocenie? Jak nowatorskie, dojrzałe, stabilne, przydatne (w porównaniu z innymi) jest to co zostało zaproponowane i stworzone. Czy przeprowadzono badanie empiryczne w środ. przem. i/lub akademickim?

Projekt (zespołowy 4 os.) - cel 2:

- Stworzenie narzędzia (vide cel 1) z wykorzystaniem ambitnej kompozycji zwinnych metodyk, praktyk i narzędzi wytwarzania oprogramowania
- Waga oceny: 50% punktów, punktacja (0-5 pkt.)
- Kiedy ocena: na zajęciach 4..13 [max.10x5pkt]
- Co podlega ocenie? Jak ambitna i dojrzała jest kompozycja zwinnych metodyk, praktyk i narzędzi wytwarzania oprogramowania (w tym infrastruktury projektowej).

Projekt - badanie empiryczne

- Grupa mająca gotowe narzędzie zgłasza je (mailem) do empirycznej oceny.
- O tym które narzędzia będą badane (do 13 tyg.) na zajęciach (udział w badaniu punktowany) decyduje prowadzący zajęcia biorąc pod uwagę:
  - dojrzałość narzędzia, istniejące ograniczenia i potencjalne walory badawcze i praktyczne
  - **kolejność zgłoszeń** (nie wszystkie narzędzia i praktyki możemy poddać badaniu na zajęciach ze względu na ograniczenia czasowe)
- Możliwa ewaluacja poza uczelnią np. w środowisku przemysłowym (szczególnie cenne).
- Wymagany szczegółowy plan i raport z badania (LaTeX).

Projekt – skala ocen i wymaganie wstępne do uzyskania zaliczenia :

Wymaganie wstępne: Student może opuścić co najwyżej 2 zajęcia projektowe.

Skala ocen: 91…100 - 5.5

81…90 – 5.0

71..80 – 4.5

61..70 – 4.0

51..60 – 3.5

41...50 – 3.0

0…40 – 2.0

Uwaga: Prowadzący zajęcia może ustalić dodatkowe wymagania i możliwość zdobycia dodatkowych punktów (np. za współpracę w przygotowaniu publikacji).

Projekt: zasady dyskwalifikacji

- Student otrzyma ocenę niedostateczną z projektu, jeżeli przedłoży rozwiązanie, którego nie jest autorem lub będące plagiatem lub kopią całości lub pewnej części innego rozwiązania (nie dotyczy np. korzystania z wzorców projektowych, frameworków itp.).

Przykładowe tematy

Narzędzia/Praktyki + narzędzia wspomagające:

- Programowanie (np. CATDD, PP), testowanie i refaktoryzację dodu
- Badanie jakości kodu,
- Badanie jakości testów (jednostkowych, akceptacyjnych)
- Estymację/predykcję czasu realizacji US.
- Zarządzanie projektem.

Ważna jest integracja z procesem/narzędziami IDE (np. Eclipse).

Seminarium

Tematem może być zwinna metodyka lub praktyka wytwarzania oprogramowania np.:

- Behaviur-Driven Development (BDD)
- Acceptance Test-Driven Development
- Agile RUP,
- Scrum,
- Lean Software Development,
- Kanban,
- XP,
- Crystal
- Tematy do uzgodnienia z prowadzącym np.:
  - propozycja własnej praktyki ;-)
  - agile w dużych projektach przemysłowych – doświadczenia

Semiarium - ocena

„You never get a second chance...”

Kryteria oceny:

- Jasno zdefiniowane cele prezentacji i korzyści dla słuchaczy (Clearly defined goals of presentation or benefits for the listeners)
- Poprawna komunikacja (Communication)
- Właściwy poziom szczegółowości prezentacji i dobór przykładów (Presentation content, level of details, examples)
- Właściwy czas prezentacji (Timing)

Seminarium – kryteria oceny

- Kolejność poruszanych tematów (Presentation flow - order of topics)
- Wciągnięcie słuchaczy (Capturing audience / listeners involvement)
- Dostępne zdalnie po poprawkach (Available online on SEnS web page after revision).
- CD (.tex, .pdf, source code examples e.g., Eclipse project)
- Aktywność podczas prezentacji (questions, answers)

Literatura

- Google
- Scholar Google
- Materiały konferencji XP, Agile.
- Czasopisma naukowe z inż. oprogramowania.
- Książki (następne slajdy)

Literatura - książki (unit testing)

- Massol, T. Husted, JUnit in Action, 2nd Ed., (JUnit jumpstart - FREE: http://www.manning.com/tahchiev/Tahchiev MEAPCH1.pdf)
- J. B. Rainsberger, JUnit Recipes: Practical Methods for Programmer Testing, Manning 2004

Literatura - książki TDD/BDD

- K. Beck, Test Driven Development: By Example, AW02
- L. Koskela, Test Driven: TDD and Acceptance TDD for Java Developers, Manning 2007
- J. Langr, Agile Java(TM): Crafting Code with Test-Driven Development (Robert C. Martin Series)
- K. Pugh, Lean-Agile Acceptance Test-Driven Development: Better Software Through Collaboration,2010
- D. Chelimsky i in., The RSpec Book: Behaviour Driven Development with Rspec, Cucumber, and Friends, Pragmatic Bookshelf 2010

Literatura - książki (XP/Scrum)

- K. Beck, Extreme Programming Explained: Embrace Change, Addison-Wesley, 2004
- M. Cohn, Succeeding with Agile: Software Development Using Scrum, AW 2009
- M. Cohn, Agile Estimating and Planning, PRH 2005
- L. Crispin, Testing Extreme Programming, AW 2002.
- H. Kniberg, Scrum and XP from the Trenches, 2007
- M. Fowler, The New Methodology,
- R. Martin, The Process – see dX: A minimal RUP process, http://www.objectmentor.com/resources/articles/RUPvsXP.pdf

Literatura - książki (refactoring)

- M. Fowler, K. Beck, J.Brant, W. Opdyke, D. Roberts, Refactoring: Improving the Design of Existing Code, AW99

Literatura - książki (design patterns)

- M. Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley 2002.
  - a wise discussion of many problems in enterprise software, healthy distance from implementation technologies such as J2EE.
  - Fowlers First Law of Distributed Objects ("Don't distribute your objects")
  - the POJO (Plain Old Java Object) term to give plain Java objects buzzword compliance.
- GoF, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley 1995 – 23 patterns
- D. Alur, J. Crupi, D. Malks, Core J2EE Patterns, PH 2003 - technology-centric "J2EE" patterns that defines a standard naming (e.g.Service Locator, Business Delegate, Front Controller)