Proiecte

Testarea Sistemelor Software

Cerințe

- **Finalitate**:
  Nota N1 - ProiectI
  Nota N2 - ProiectII
  Fiecare notă trebuie să fie >= 5 (cu alte cuvinte, fiecare proiect este obligatoriu!)
  Media finală = (N1+N2)/2
- **Proiectul I este individual** şi va fi prezentat în cadrul laboratorului. Se va aduce în **format printat + electronic** (trimis pe email la adresa sorina.predut@gmail.com).
- **Proiectul II se va realiza în echipă** şi poate fi prezentat în timpul semestrului conform unei programări stabilite pentru fiecare echipă în parte (prezentarea acestuia se va face începând cu 27.03.2017). **Componenţa echipei şi tema aleasă vor fi trimise pe email la adresa** sorina.predut@gmail.com **până la data de 6.03.2017.**
- **Proiectul II poate fi prezentat şi în sesiune, în ziua examenului. Dacă în componenţa echipei sunt membri din grupe diferite se va considera ID-ul cel mai mic dintre numerele grupelor şi proiectul va fi predat în ziua în care este programat examenul pentru acea grupă.** Fiecare membru al echipei trebuie să prezinte contribuţia sa la proiect şi să răspundă la întrebări legate de întreg proiectul, în ziua în care este predat sau în ziua în care are examen grupa sa, dacă în acea zi examenul de la grupa cu ID minim se suprapune cu altul de la grupa sa.
- Pentru realizarea proiectului II se pot forma echipe de **maxim 5 studenți**. Fiecare echipă va alege una dintre temele precizate. O temă poate fi aleasă de cel mult trei echipe.
- **Prezentarea proiectului II va fi sub formă de slide-uri** şi va dura aproximativ 30 - 40 minute. Acolo unde este cazul, prezentarea va fi însoțită de un demo pentru implementarea realizată. La sfârşitul prezentării se va preciza (ca procentaj) contribuţia fiecărui membru al echipei la realizarea acesteia. Slide-urile au ca scop evidenţierea ideilor principale prezentate şi facilitarea înţelegerii acestora. Din acest motiv, acestea nu vor conţine text nestructurat şi detalii nesemnificative.
- Proiectul cu toate detaliile semnificative va fi predat în **format printat + electronic** (trimis pe email).

Proiect I: JUnit + functional & structural testing + mutation testing

Să se scrie un program Java, precum şi cerințele (specificația) acestuia.

1. Pe baza cerințelor programului, să se genereze date de test folosind metode funcționale: (a) partiționare în clase de echivalenţă, (b) analiza valorilor de frontieră, (c) partiționare în categorii. Să se implementeze cele 3 seturi de test obținute folosind JUnit.
2. Să se transforme programul într-un graf orientat şi, pe baza acestuia, să se genereze date de test care realizează acoperire la nivel de (a) instrucțiune, (b) decizie, (c) condiție. Să se implementeze cele 3 seturi de test folosind JUnit.
3. Să se calculeze complexitatea programului folosind formula lui McCabe, precum şi numărul de circuite independente.
4. Să se propună un set de teste care să realizeze o acoperire la nivel de cale şi să se implementeze setul de test rezultat în JUnit.

5. Să se folosească un generator de mutanți (e.g. MuJava/Muclipse sau alt utilitar pentru a crea mutanți pentru programul dat.
6. Să se ruleze seturile de test de la punctele 1), 2) si 4) contra mutanților generați şi să se explice rezultatele.
7. Să se creeze teste suplimentare pentru a omorî 2 dintre mutanții neechivalenți rămași în viață.

Referințe: [1-7].

Proiect II

Tema 1: Search-based software testing

Se consideră următoarea problemă: date fiind un program şi o cale în graful asociat programului, se urmărește generarea unor date de test ce execută această cale.

- Să se prezinte rezolvarea problemei folosind algoritmi genetici. Să se explice forma funcției de fitness folosite.
- Să se implementeze un algoritm care rezolvă problema de mai sus şi să se ilustreze funcţionarea sa pentru un program dat. Să se comenteze rezultatele, sugerând îmbunătățiri.

Referinţe: [8-11].

Tema 2: Search-based software testing

- Să se prezinte metoda Chaining Approach pentru generare de date de test. Să se explice utilitatea metodei.
- Să se implementeze algoritmul şi să se ilustreze funcționarea sa pentru un program dat.

Referințe: [12].

Tema 3: Finite state machine based testing

- Să se prezinte metodele W şi Wp de generare de teste pe baza unei specificații de forma unui FSM (Finite State Machine). Să se descrie condiţiile de aplicare a metodelor. Să se explice, pe un exemplu, aplicarea metodei pentru diagrama de stare a unei clase.
- Să se implementeze metoda W.

Referințe: [13].

Tema 4: Finite state machine based testing

- Să se prezinte conceptul de secvență UIO (Unique Input/Output) şi relevanţa acestuia pentru testarea de conformanţă. Să se prezinte modul de construcție a unei secvențe UIO folosind un State Splitting Tree.

- Să se prezinte modul de construcție a unei secvențe UIO folosind un algoritm genetic.
- Să se implementeze algoritmul genetic şi să se evalueze funcționarea acestuia.

Referințe: [15].

Tema 5: Code coverage and mutation testing

- Să se realizeze un studiu comparativ a cel puţin 3 code coverage tools, evidențiindu-se utilitatea şi ușurința în folosire a fiecăruia. Pe baza unor exemple de cod, se vor evidenția diferențele dintre tool-uri.
- Să se realizeze un studiu comparativ a cel puţin 3 mutation tools, evidențiindu-se utilitatea şi ușurința în folosire a fiecăruia. Pe baza unor exemple de cod, se vor compara operatorii de mutație ai fiecăruia, evidențiindu-se utilitatea acestor operatori pentru evaluarea testelor şi detectarea erorilor.
- Să se evidențieze, folosind exemple, legătura dintre code coverage şi mutation testing.

Notă: nu se ia în considerare traducerea instrucțiunilor de utilizare a utilitarelor respective.

Referințe: [4-7, 14, 17, 18].

Tema 6: Model based Testing and Analysis with C# and NModel

- Prezentați utilizarea mpv (Model Program Viewer) pentru vizualizarea şi explorarea modelului şi a scenariilor.
- Prezentați mpv pentru analiza modelului (safety, liveness, invarianţi, stări acceptoare, stări moarte)
- Conceptele prezentate vor fi ilustrate pe exemple proprii (create de echipă). Pentru a evidenția modul de explorare a modelului cu stări (FSM), folosiți mpv pe unul sau mai multe exemple cu parametrul MaxTransitions suficient de scăzut astfel încât modelul să nu fie vizualizat în totalitate.

Referințe: [16] Capitolul 6, [22].

Tema 7: Model based Testing and Analysis with C# and NModel

- Prezentați facilităţile NModel pentru structurarea modelelor: features şi composition.
- Conceptele prezentate vor fi ilustrate pe exemple proprii (create de echipă).

Referințe: [16] Capitolul 7, [22].

Tema 8: Model based Testing and Analysis with C# and NModel

- Prezentați facilităţile NModel pentru vizualizarea şi explorarea modelelor cu număr mare (potențial infinit) de stări şi tranziții.
- Prezentați atributele Domain, Feature şi utilitatea acestora.

- Prezentați tehnicile de reducere a modelului (pruning techniques).
- Conceptele prezentate vor fi ilustrate pe exemple proprii (create de echipă).

Referințe: [16] Capitolul 11, [22].

Tema 9: Model based Testing and Analysis with C# and NModel

- Prezentați generatorul de teste otg (Offline Test Generator).
- Prezentați rolul şi modul de utilizare pentru test harness şi stepper.
- Prezentați execuția de teste folosind ct (Conformance Tester).
- Prezentați limitările acestor utilitare, exemple de erori nedetectate. Explicați aceste limitări.
- Utilitarele şi conceptele prezentate vor fi ilustrate pe exemple proprii (create de echipă).

Referințe: [16] Capitolul 8, [22].

Tema 10: Model based Testing with GraphWalker

- Prezentați utilitarul de generare de teste GraphWalker din modele de formă (Extended) Finite State Machines.
- Prezentați generatoarele şi criteriile de oprire (stop criteria) disponibile. Ilustrați cu exemple adecvate.
- Prezentați şi ilustrați cu exemple generarea online de teste (online test sequence generation).
- Utilitarele şi conceptele prezentate vor fi ilustrate pe exemple proprii (create de echipă).

Referințe: [19]

Tema 11: Model based Testing and Analysis with PyModel

- Prezentați utilitarele pma.py şi pmg.py pentru generarea şi vizualizarea unui FSM (Finite State Machine) pe baza unui program model.
- Să se prezinte strategii pentru limitarea explorării unui program infinit.
- Să se prezinte modul de verificare a unor condiții de safety şi liveness.
- Utilitarele şi conceptele prezentate vor fi ilustrate pe exemple proprii (create de echipă).

Referințe: [20, 21], [16] Capitolul 6.

Tema 12: Model based Testing and Analysis with PyModel

- Prezentați utilitarul de generare şi execuție de teste pmt.py.
- Prezentați strategiile de generare de teste disponibile.
- Descrieți importanţa compunerii automatelor în generarea de teste şi validarea modelelor.
- Prezentați modurile de testare offline şi on-the-fly, precum şi diferența dintre ele.
- Prezentați execuția testelor folosind un test harness (stepper).
- Utilitarele şi conceptele prezentate vor fi ilustrate pe exemple proprii (create de echipă).

Referințe: [20, 21], [16] Capitolul 8.

Alte teme:

Echipele pot propune (şi trimite spre acceptare) teme similare, e.g.

- Metode şi algoritmi de model based testing / test generation / search based testing
- Tool-uri pentru model based testing and analysis

Bibliografie

[1] JUnit slides (material de curs)
[2] Functional tesing (material de curs)
[3] Structural testing (material de curs)
[4] Mutation testing (material de curs)
[5] MuJava home page http://cs.gmu.edu/~offutt/mujava/
[6] MuClipse homepage http://muclipse.sourceforge.net/
[7] Mutation testing repository: http://crestweb.cs.ucl.ac.uk/resources/mutation_testing_repository/
[8] Phil McMinn: Search-based software test data generation: a survey. Softw. Test., Verif. Reliab. 14(2): 105-156 (2004)
[9] Nigel Tracey, John Clark, John McDermid and Keith Mander. A search-based automated test-data generation framework for safety-critical systems. In Systems engineering for business process change: new directions, 2002, 174-213, Springer-Verlag New York, New York, NY, USA.
[10] Genetic algorithm tools: http://jgap.sourceforge.net/, http://www.jaga.org/, http://cs.gmu.edu/~eclab/projects/ecj/
[11] Raluca Lefticaru, Florentin Ipate: Automatic State-Based Test Generation Using Genetic Algorithms. In 9th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2007): 188-195, IEEE Computer Society, 2007.
[12] R. Ferguson and B. Korel. The chaining approach for software test data generation. ACM Transactions on Software Engineering and Methodology, 5(1):63-86, 1996.
[13] Aditya P. Mathur. Foundations of Software Testing, Pearson Education 2008. Chapter 6: Test Generation: FSM Models.
[14] Yue Jia and Mark Harman: An Analysis and Survey of the Development of Mutation Testing, IEEE Transactions on Software Engineering 37(5): 649-678, 2011.
[15] Qiang Guo, Robert M. Hierons, Mark Harman, Karnig Derderian. Computing unique input/output sequences using genetic algorithms. In Proceedings of the 3rd International Workshop on Formal Approaches to Testing of Software (FATES'2003), volume 2931 of LNCS, 2004.
[16] Jonathan Jacky, Margus Veanes, Colin Campbell, Wolfram Schulte. Model-based Software Testing and Analysis with C#. Cambridge University Press, 2008.
[17] Qian Yang, J. Jenny Li, David M. Weiss: A Survey of Coverage-Based Testing Tools. The Computer Journal 52(5): 589-597 (2009)
[18] Muhammad Shahid, Suhaimi Ibrahim, Naz'ri Mohd Mahrin, An Evaluation of Test Coverage Tools in Software Testing, Proceedings of the International Conference on Telecom Technology and Applications (ICTTA 2011), Thomson ISI, Sydney, Australia, May 2-4, 2011.
[19] http://graphwalker.org/
[20] http://staff.washington.edu/jon/pymodel/www/

[21] Jonathan Jacky. PyModel: Model-based testing in Python. http://staff.washington.edu/jon/pymodel/talks/pymodel-scipy2011.pdf
[22] http://nmodel.codeplex.com/

ID proiect	Titlu	Echipa
1	Search-based software testing	
2	Search-based software testing - Chaining Approach	
3	Finite state machine based testing	
4	Finite state machine based testing	
5	Code coverage and mutation testing	
6	Model based Testing and Analysis with C# and NModel	
7	Model based Testing and Analysis with C# and NModel	
8	Model based Testing and Analysis with C# and NModel	
9	Model based Testing and Analysis with C# and NModel	
10	Model based Testing with GraphWalker	
11	Model based Testing and Analysis with PyModel	
12	Model based Testing and Analysis with PyModel	
13		
14		
15		
16		
17		
18		
19		
20		