

Testare si Verificare, Master IS, an 2, sem 2
An univ. 2021-2022
Curs: Florentin Ipathe
Laborator: Ana Iova (Țurlea)

I. Obiective: Asimilarea principalelor concepte si tehnici de testare a sistemelor software, model checking si model-based testing.

II. Cerinte

Notarea se va face pe baza proiectului de laborator (50%) si proiectului de curs (50%).

Proiectul de laborator va fi individual si va fi prezentat la laborator conform programarii stabilite cu cadrul didactic de la laborator.

Proiectele de curs se vor efectua in echipe (dimensiunea echipei pentru fiecare tema este specificata mai jos) si vor fi prezentate la curs in saptamanile 3-10 conform programarii stabilite, astfel incat numarul maxim de studenti care prezinta in cadrul unui curs sa fie 10. Tema proiectului de curs va fi aleasa din lista de mai jos. Fiecare tema din lista poate fi aleasa de cel mult 3 echipe.

Prezentarea proiectului de curs va fi sub forma de slide-uri, insotite de demo-uri. La prezentare este necesara prezenta intregii echipe, fiecare student descriind principala sa contributie la proiect. Timpul alocat fiecarui student este de 10 minute. Atunci cand echipa considera ca nu toti membrii echipei au avut o contributie egala la realizarea proiectului, se va indica, in procente, contributia estimata a fiecaruia.

Proiect alternativ: Studentii care sunt interesati de un doctorat in domeniile ingineriei software, testarii si verificarii, machine learning sau bioinformaticii pot lua legatura cu cadrul didactic de curs pentru stabilirea unui proiect alternativ, a carui tema sa poata fi continuata in cadrul studiilor doctorale. Pentru informatii suplimentare se poate consulta <http://www.ifsoft.ro/~florentin.ipate/>.

III. Proiect laborator

Sa se scrie un program in Java, precum si cerintele (specificatia) acestuia.

1. Pe baza cerintelor programului, sa se genereze date de test folosind a) equivalence partitioning b) boundary value analysis si c) cause-effect graphing. Sa se implementeze testele obtinute folosind JUnit.
2. Sa se stabileasca nivelul de acoperire realizat de fiecare dintre seturile de teste de la 1) a), b) si c) folosind unul dintre utilitarele de code coverage prezentate in [6].
3. Sa se transforme programul intr-un graf orientat si, pe baza acestuia, sa se gaseasca un set de teste care satisface criteriul modified condition/decision coverage (MC/DC).
4. Sa se scrie un mutant de ordinul 1 echivalent al programului.
5. Pentru unul dintre cazurile de testare de mai sus sa se scrie un mutant ne-echivalent care sa fie omorat de catre test si un mutant ne-echivalent care sa nu fie omorat de catre test.

Referinte: [1-7]

IV. Teme proiecte curs

1. FSM based testing

- Prezentați problematica și principalele tehnici de FSM based testing: W-method, UIO, DS.
- Scrieți un program care implementează W-method și ilustrați aplicarea acesteia pe un exemplu.

Marime echipa: 3 studenți

Bibliografie: [8]

2. Search based testing pentru EFSM

- Prezentați metoda de generare de date de test prezentată în [10].
- Scrieți un program care implementează metoda de generare de date de test prezentată în [10] și ilustrați funcționarea acesteia pe un exemplu.

Marime echipa: 2 studenți

Bibliografie: [9]

3. Automatic Test Suite Generation cu EvoSuite

- Prezentați utilitarul EvoSuite, care generează automat cazuri de testare cu aserțiuni pentru clase scrise în cod Java folosind o abordare hibridă pentru generarea și optimizarea întregii suite de testare ce satisface un criteriu de acoperire.
- Ilustrați capabilitățile utilitarului folosind unul sau mai multe exemple create de echipa.

Marime echipa: 2 studenți

Bibliografie: [10,11]

4. Model based testing cu JSXM

- Prezentați limbajul de modelare bazat pe stream X-machines precum și principalele facilități oferite de utilitarul JSXM: animarea modelului, generarea testelor pe baza modelului de forma stream X-machine, implementarea testelor în JUnit.
- Ilustrați aceste capabilități folosind unul sau mai multe exemple create de echipa.

Marime echipa: 3 studenți

Bibliografie: [12-14]

5. Model based testing for Cloud Services cu Broker@Cloud

- Prezentați limbajul de modelare bazat pe stream X-machines precum și principalele facilități oferite de utilitarul Broker@Cloud verification and testing tool: verificare, test generation și test grounding.
- Ilustrați aceste capabilități folosind unul sau mai multe exemple create de echipa.

Marime echipa: 3 studenți

Bibliografie: [15-16]

6. Model based testing cu GraphWalker

- Prezentați principalele facilități oferite de utilitarul GraphWalker: generarea testelor din modelul mașinilor cu stări finite (FSM) folosind algoritmi de traversare precum A* sau parcurgere random cu diverse criterii de acoperire (state, edge, requirement).
- Ilustrați aceste capabilități folosind unul sau mai multe exemple create de echipa.

Marime echipa: 2 studenți

Bibliografie: [17,18]

7. Modelare si analiza cu Event-B, Rodin

- Prezentati capabilitatile de modelare si analiza ale limbajului Event-B: evenimente, rafinare, demonstrare automata a proprietatilor.
- Ilustrati aceste capabilitati folosind unul sau mai multe exemple create de echipa.

Marime echipa: 3 studenti

Bibliografie: [19,20]

8. Model checking cu Event-B, Rodin si Pro B

- Prezentati abordarea de model checking folosind Event-B, Rodin si Pro B descrisa in [21].
- Ilustrati aceasta abordare folosind unul sau mai multe exemple create de echipa (diferite de exemplul din articolul [21]).

Marime echipa: 3 studenti

Bibliografie: [20-22]

9. Model checking cu NuSMV

- Prezentati pe scurt logica LTL si logica CTL.
- Ilustrati capabilitatile de model checking (verificare de proprietati LTL si CTL) folosind unul sau mai multe exemple create de echipa.

Marime echipa: 3 studenti

Bibliografie: [23,24]

10. Modelare, simulare si verificare cu membrane systems si kPWorkbench

- Prezentati formalismul kP systems si capabilitatile de modelare si verificare formala a suitei kPWorkbench.
- Ilustrati aceste capabilitati folosind unul sau mai multe exemple create de echipa.

Marime echipa: 3 studenti

Bibliografie: [25-27]

V. Bibliografie

- [1]. Functional testing, material de curs.
- [2]. Structural testing, material de curs.
- [3]. Mutation testing, material de curs.
- [4]. <http://www.vogella.com/tutorials/JUnit/article.html>
- [5]. <https://cs.gmu.edu/~offutt/rsrch/mut.html>
- [6]. <https://www.softwaretestinghelp.com/code-coverage-tools/>
- [7]. <https://cs.gmu.edu/~offutt/mujava/>
- [8]. Aditya P. Mathur. Foundations of Software Testing, Pearson Education 2008. Chapter 6: Test Generation: FSM Models.
- [9]. Raluca Lefticaru, Florentin Ipatu: Automatic State-Based Test Generation Using Genetic Algorithms, in SYNASC, 188-195 (2007).
<http://www.ifsoft.ro/~florentin.ipatu/publications/SYNASC%202007%20Automatic%20State-Based%20Test%20Generation%20Using%20Genetic%20Algorithms.pdf>
- [10]. <https://www.evosuite.org/>
- [11]. Gordon Fraser, Andrea Arcuri: EvoSuite: automatic test suite generation for object-oriented software. SIGSOFT FSE 2011: 416-419
https://www.researchgate.net/profile/Andrea-Arcuri-3/publication/221560749_EvoSuite_Automatic_test_suite_generation_for_object-oriented_software/links/5491110c0cf2747c208b4d0c/EvoSuite-Automatic-test-suite-generation-for-object-oriented-software.pdf

[oriented_software/links/595cd3e50f7e9b3aefaddbd0/EvoSuite-Automatic-test-suite-generation-for-object-oriented-software.pdf](http://www.ifsoft.ro/~florentin.ipate/publications/STVR98%20-%20Specification%20and%20testing%20using%20generalised%20machines.pdf)

- [12]. Florentin Ipaté, Mike Holcombe: Specification and testing using generalized machines: a presentation and a case study, *Software Testing, Verification and Reliability*, 8, 61-81 (1998) <http://www.ifsoft.ro/~florentin.ipate/publications/STVR98%20-%20Specification%20and%20testing%20using%20generalised%20machines.pdf>
- [13]. <http://www.jsxm.org/>
- [14]. Dimitris Dranidis, Konstantinos Bratanis, Florentin Ipaté: JSXM: a tool for automated test generation, in *SEFM*, 352-366 (2012).
http://www.ifsoft.ro/~florentin.ipate/publications/SEFM2012_CR.pdf
- [15]. <http://staffwww.dcs.shef.ac.uk/people/A.Simons/broker/>
- [16]. Lefticaru, R. and Simons, A.J.H. (2015) X-Machine Based Testing for Cloud Services. In: Ortiz, G. and Tran, C., (eds.) *Advances in Service-Oriented and Cloud Computing. Workshops of ESOC 2014*, September 2-4, 2014, Manchester, UK. *Lecture Notes in Computer Science*, 508 . Springer , pp. 175-189.
<https://eprints.whiterose.ac.uk/98321/1/XMtestingforcloud.pdf>
- [17]. <https://graphwalker.github.io/>
- [18]. Muhammad Nouman Zafar, Wasif Afzal, Eduard Enoiu, Athanasios Stratis, Aitor Arrieta, Goiuria Sagardui: Model-Based Testing in Practice: An Industrial Case Study using GraphWalker. *ISEC 2021*: 5:1-5:11
http://ebiltegia.mondragon.edu/xmlui/bitstream/handle/20.500.11984/5411/Model-Based%20Testing%20in%20Practice_An%20Industrial%20Case%20%20Study%20using%20GraphWalker.pdf?sequence=1&isAllowed=y
- [19]. Jean-Raymond Abrial. *Modeling in Event-B: System and Software Engineering*.
- [20]. <http://www.event-b.org/>
- [21]. Adrian Turcanu, Talal Shaikh and Cristina Nicoleta Mazilu, On Model Checking of a Robotic Mechanism. <https://scholars.direct/Articles/robotics/jra-4-018.pdf>
- [22]. <https://www3.hhu.de/stups/prob/>
- [23]. <https://www.cs.utexas.edu/users/moore/acl2/seminar/2010.05-19-krug/slides.pdf>
- [24]. <https://nusmv.fbk.eu/>
- [25]. <https://github.com/Kernel-P-Systems/kPWorkbench>
- [26]. R. Lefticaru, M. E. Bakir, S. Konur, M. Stannett, F. Ipaté: Modelling and Validating an Engineering Application in Kernel P Systems, *Int. Conf. on Membrane Computing 2017, LNCS*, 183-195, 2017. <http://www.ifsoft.ro/~florentin.ipate/publications/EngApl-CMC2017.pdf>
- [27]. S. Konur, L. Mierla, F. Ipaté, M. Gheorghe: kworkbench: A software suit for membrane systems, *SoftwareX*, 11: 100407, 2020.
<http://www.ifsoft.ro/~florentin.ipate/publications/SoftwareXKMIG.pdf>