Curs **Testare si Verificare**, Master IS, an 2, sem 2
An univ. 2018-2019
Curs: Florentin Ipate
Laborator: Ana 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 de proiect laborator (50%) si proiect 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 3-9, conform programarii stabilite, astfel incat sa nu fie mai mult de 3 prezentari pe curs. Tema proiectului de curs va fi aleasa din lista de mai jos. Fiecare tema din lista poate fi aleasa de cel mult o echipa, cu exceptia temelor 1, 3, 4, 5 si 9, care pot fi alese de 2 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. 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, metodelor formale sau bioinformaticii pot lua legatura cu cadrul didactic de curs pentru stabilirea unui proiect alternativ (in locul celor doua proiecte), a carui tema sa poata fi continuata in cadrul studiilor doctorale. Pentru o lista neexhaustiva a unor directii de cercetare posibile se poate consulta http://www.ifsoft.ro/~florentin.ipate/.

III. Proiect laborator

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

1. Pe baza cerintelor programului, sa se genereze date de test folosind a) equivalence partitioning si b) boundary value analysis. Sa se implementeze testele obtinute folosind JUnit.
2. Sa se transforme programul intr-un graf orientat si, pe baza acestuia, sa se genereze date de test care realizeaza acoperire la nivel de a) instructiune, b) decizie, c) conditie d) modified condition/decision. Sa se implementeze testele obtinute folosind JUnit.
3. Sa se calculeze complexitatea programului folosind formula lui McCabe, precum si numarul de circuite independente.
4. Sa se propuna un set de teste pentru o acoperire la nivel de cale si sa se implementeze testele rezultate in JUnit.
5. Sa se foloseasca un generator de mutanti (e.g. PIT, MuJava/Muclipse) pentru a crea mutanti pentru programul dat. Sa se ruleze seturile de test de la punctele 1), 2) si 4) pe mutantii generati si sa se explice rezultatele.
6. Sa se creeze teste suplimentare pentru a omora 2 dintre mutantii neechivalenti ramasi in viata.

Referinte: [1-7]

IV. Teme proiecte curs

1. **FSM based testing**
   Bibliografie: [8]
   Marime echipa: 4 studenti
   Durata prezentare: aprox. 40 min (4 x 10 min)
   Raportul va prezenta problematica si principalele tehnici de FSM based testing (W-method, UIO, DS, Wp-method), precum si algoritmii pentru constructia seturilor de testare. Se va implementa W-method si va fi prezentata functionarea acesteia pe un exemplu.

2. **Search based testing**
   Bibliografie: [9]
   Marime echipa: 3 studenti
   Durata prezentare: aprox. 30 min (3 x 10 min)
   Raportul va prezenta problematica si principalele tehnici de search based testing. Se va implementa unul dintre algoritmii de search based testing prezentati si va fi prezentata functionarea acestuia pe un exemplu.

3. **Event-B and Pro B**
   Bibliografie: [10] Chapter 2, [11,12].
   Marime echipa: 4 studenti
   Durata prezentare: aprox. 40 min (4 x 10 min)
   Sa se ilustreze capabilitatile de modelare si analiza ale limbajului Event-B (evenimente, rafinare, demonstrare automata a proprietatilor) pe unul sau mai multe exemple create de echipa. Sa se ilustreze folosirea Pro B pentru animarea modelului si model checking.

4. **NuSMV**
   Bibliografie: [13].
   Marime echipa: 4 studenti
   Durata prezentare: aprox. 40 min (4 x 10 min)
   Sa se ilustreze capabilitatile de model checking (verificare de proprietati LTL si CTL) pe unul sau mai multe exemple create de echipa.

5. **NModel Analysis**
   Bibliografie: [14], capitolele 6, 7, 11.
   Marime echipa: 4 studenti
   Durata prezentare: aprox. 40 min (4 x 10 min)
   Raportul va prezenta, sintetic, principalele facilitati oferite de utilitar pentru analiza modelului (safety, liveness, invarianti, stari acceptoare, stari moarte), pentru structurarea modelelor (features, composition), pentru vizualizarea si analiza modelelor cu numar mare (potential infinit) de stari si tranzitii (Domain, Feature, pruning techniques) si pentru testare (Offline Test Generator, Conformance Tester). Toate acestea vor fi ilustrate cu unul sau mai multe studii de caz create de catre echipa.

6. **NModel Testing**
   Bibliografie: [14], capitolul 8.
   Marime echipa: 2 studenti
   Durata prezentare: aprox. 20 min (2 x 10 min)
   Raportul va prezenta, sintetic, principalele facilitati oferite de utilitar pentru analiza modelului (safety, liveness, invarianti, stari acceptoare, stari moarte), pentru structurarea modelelor (features, composition), pentru vizualizarea si analiza modelelor cu numar mare

(potential infinit) de stari si tranzitii (Domain, Feature, pruning techniques) si pentru testare (Offline Test Generator, Conformance Tester). Toate acestea vor fi ilustrate cu unul sau mai multe studii de caz create de catre echipa.

7. **fMBT(free Model-based Testing)**
   Bibliografie: [15]
   Marime echipa: 2 studenti
   Durata prezentare: aprox. 20 min (2 x 10 min)
   Raportul va prezenta, sintetic, principalele facilitati oferite de utilitar: generarea cazurilor de testare pentru modele scrise in limbajul preconditii/postconditii AAL/Python folosind diverse euristici: random, weighted random, lookahead, etc. Toate acestea vor fi ilustrate cu unul sau mai multe studii de caz create de catre echipa.

8. **GraphWalker**
   Bibliografie: [16]
   Marime echipa: 2 studenti
   Durata prezentare: aprox. 20 min (2 x 10 min)

Raportul va prezenta, sintetic, principalele facilitati oferite de utilitar: generarea testelor din modelul masinilor cu stari finite (FSM) folosind algoritmi de traversare precum A* sau parcurgere random cu diverse criterii de acoperire (state, edge, requirement). Toate acestea vor fi ilustrate cu unul sau mai multe studii de caz create de catre echipa.

9. **JSXM**
   Bibliografie: [17, 18]
   Marime echipa: 4 studenti
   Durata prezentare: aprox. 40 min (4 x 10 min)

Raportul va prezenta, sintetic, limbajul de modelare bazat pe stream X-machines precum si principalele facilitati oferite de utilitarul JSXM: animarea modelului, generarea testelor pe baza modelului de forma stream X-machine, implementarea testelor in JUnit. Toate acestea vor fi ilustrate cu unul sau mai multe studii de caz create de catre echipa.

10. **Modbat**
    Bibliografie: [19,20]
    Marime echipa: 3 studenti
    Durata prezentare: aprox. 30 min (3 x 10 min)

Raportul va prezenta, sintetic, principalele facilitati oferite de utilitarul de model-based testing, bazat pe masini cu stari finite (extinse) adnotate. Va fi prezentat si limbajul de modelare specific domeniului, cu caracteristici precum tranzitii probabiliste si ne-deterministe, mostenire si exceptii. Toate acestea vor fi ilustrate cu unul sau mai multe studii de caz create de catre echipa.

11. **ModelJUnit**
    Bibliografie: [21,22]
    Marime echipa: 2 studenti
    Durata prezentare: aprox. 20 min (2 x 10 min)

Raportul va prezenta, sintetic, principalele facilitati oferite de utilitar: realizarea modelelor de forma masinilor cu stari finite (FSM) sau masinilor cu stari finite extinse (EFSM), generarea testelor pe baza acestor modele si masurarea diverselor metrici de acoperire. Toate acestea vor fi ilustrate cu unul sau mai multe studii de caz create de catre echipa.

12. **OSMO**
    Bibliografie: [23]
    Marime echipa: 2 studenti

Durata prezentare: aprox. 20 min (2 x 10 min)

Raportul va prezenta, sintetic, principalele facilitati oferite de utilitar: crearea si utilizarea modelelor in Java pentru generarea de teste folosind diverse strategii (random, pe baza de constrangeri pentru ghidare, etc.). Toate acestea vor fi ilustrate cu unul sau mai multe studii de caz create de catre echipa.

13. **Tcases**

Bibliografie: [24]

Marime echipa: 2 studenti

Durata prezentare: aprox. 20 min (2 x 10 min)

Raportul va prezenta, sintetic, principalele facilitati oferite de utilitar: definirea sistemului si a nivelului de acoperire in format XMP precum si generarea de teste pe baza acestor fisiere. Toate acestea vor fi ilustrate cu unul sau mai multe studii de caz create de catre echipa.

14. **PyMODEL**

Bibliografie: [25,26]

Marime echipa: 2 studenti

Durata prezentare: aprox. 20 min (2 x 10 min)

Raportul va prezenta, sintetic, principalele facilitati oferite de utilitar: offline si on-the fly testing, compunere pentru controlul scenariilor, acoperire ghidata de o cautare programabila. Toate acestea vor fi ilustrate cu unul sau mai multe studii de caz create de catre echipa.

15. **MoMuT**

Bibliografie: [27,28]

Marime echipa: 2 studenti

Durata prezentare: aprox. 20 min (2 x 10 min)

Raportul va prezenta, sintetic, principalele facilitati oferite de utilitar: testare bazata pe modele UML si generare de cazuri de testare folosind operatori de mutatie. Toate acestea vor fi ilustrate cu unul sau mai multe studii de caz create de catre echipa.

16. **JTorX**

Bibliografie: [29]

Marime echipa: 2 studenti

Durata prezentare: aprox. 20 min (2 x 10 min)

Raportul va prezenta, sintetic, principalele facilitati oferite de utilitar: testare bazata pe model pentru Labelled Transition Systems (LTS) si testarea on-the-fly a implementarii. Toate acestea vor fi ilustrate cu unul sau mai multe studii de caz create de catre echipa.

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]. http://pitest.org/

[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]. Phil McMinn: Search-based software test data generation: a survey. Softw. Test., Verif. Reliab. 14(2): 105-156 (2004)
[10]. Jean-Raymond Abrial. Modeling in Event-B: System and Software Engineering.
[11]. http://www.event-b.org/
[12]. https://www3.hhu.de/stups/prob/
[13]. http://nusmv.fbk.eu/
[14]. Jonathan Jacky, Margus Veanes, Colin Campbell, Wolfram Schulte. Model-based Software Testing and Analysis with C#. Cambridge University Press, 2008.
[15]. https://01.org/fmbt
[16]. http://graphwalker.github.io/
[17]. http://www.jsxm.org/
[18]. http://www.ifsoft.ro/~florentin.ipate/publications/SEFM2012_CR.pdf
[19]. https://people.kth.se/~artho/modbat/
[20]. http://fmv.jku.at/seidl/papers/ase15.pdf
[21]. https://sourceforge.net/projects/modeljunit/
[22]. http://www.cse.chalmers.se/edu/year/2017/course/DAT261/tutorials/modeljunit.html
[23]. https://github.com/mukatee/osmo
[24]. https://github.com/Cornutum/tcases
[25]. http://staff.washington.edu/jon/pymodel/www/
[26]. http://staff.washington.edu/jon/pymodel/talks/pymodel-scipy2011.pdf
[27]. https://momut.org/
[28]. http://www.ist.tugraz.at/aichernig/publications/papers/icst15-tools.pdf
[29]. https://fmttools.ewi.utwente.nl/redmine//projects/jtorx/wiki/