Teme laborator Testarea sistemelor software

Cerințe

- Pentru realizarea unei teme se pot forma echipe de maxim 5 studenți. Este de preferat ca temele să fie realizate în echipă, dar nu obligatoriu.
  Fiecare echipă va alege una dintre temele precizate mai jos. O temă poate fi aleasă de mai multe echipe, dar va fi verificată cu un soft antiplagiat.
- Prima pagină din documentație (fișier pdf/docx) trebuie să conțină componența echipei și tema aleasă. Un singur membru din echipă va trimite rezolvarea. Fiecare membru va primi același punctaj, indiferent de contribuția sa.
- Tehnoredactarea textului va fi făcută cu fontul Times New Roman cu caractere de 12 puncte tipografice (cu excepția secvențelor de cod sursă), având spațiere verticală la 1,5 rânduri și aliniere Justify.
- Secvențele de cod sursă vor fi integrate în document folosind o celulă (tabel cu o linie și o coloană). Nu se acceptă capturi de ecran ale codului.
  Pentru textul din celulă se va folosi fontul Courier New/Consolas cu caractere de 11 puncte tipografice, având spațiere verticală la 1 rând ca în exemplul:

```
def test_sum():
    assert sum([1, 2, 3]) == 6, "Should be 6"
```

- Pentru fiecare cerință se va crea o secțiune cu un titlu scurt care va fi adăugat automat în Cuprins.
- Grafurile vor fi realizate cu un utilitar (https://app.diagrams.net, https://www.lucidchart.com/pages/, yEd, Microsoft Visio et al.). Nu se acceptă imagini fotografiate/scanate.
- Pot fi utilizate alte referințe decât cele indicate: documentații oficiale ale tool-urilor, articole științifice și cărți (https://dblp.uni-trier.de, https://scholar.google.ro).
  Articolele științifice și cărți care nu sunt free pot fi găsite la https://sci-hub.st, http://libgen.st
- Tema (documentația) va fi predată în **format electronic până la finalul semestrului (5 mai 2023, 23:59).**
- **Link upload:** https://forms.gle/B8E2tWw7F9qsKfTQ9

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 funcționarea sa pentru un program dat. Să se comenteze rezultatele, sugerând îmbunătățiri.

Referinţe:

[1] Phil McMinn, Search-based software test data generation: a survey. Softw. Test., Verif. Reliab. 14(2): 105-156 (2004)

[2] 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.
[3] Genetic algorithm tools: http://jgap.sourceforge.net/, http://cs.gmu.edu/~eclab/projects/ecj/
[4] 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.

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:

[1] 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.

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 condițiile 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:

[1] Aditya P. Mathur, Foundations of Software Testing, Pearson Education 2008. Chapter 6: Test Generation: FSM Models.

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:

[1] 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.

Tema 5: Code Coverage and Mutation Testing

- Să se realizeze un studiu comparativ a cel puţin 3 code coverage tools, evidenţiindu-se utilitatea şi uşurinţa în folosire a fiecăruia. Pe baza unor exemple de cod, se vor evidenţia diferenţele dintre tool-uri.
- Să se realizeze un studiu comparativ a cel puţin 3 mutation tools, evidenţiindu-se utilitatea şi uşurinţa în folosire a fiecăruia. Pe baza unor exemple de cod, se vor compara operatorii de mutaţie ai fiecăruia, evidenţiindu-se utilitatea acestor operatori pentru evaluarea testelor şi detectarea erorilor.
- Să se evidenţieze, folosind exemple, legătura dintre code coverage şi mutation testing.

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

Referinţe:

[1] Tutoriale testare unitară
JUnit: https://www.vogella.com/tutorials/JUnit/article.html
MSTest:
https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-mstest
https://docs.microsoft.com/en-us/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code?view=vs-2019
NUnit: https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-nunit
PyUnit:
https://realpython.com/python-testing/
https://docs.python-guide.org/writing/tests/

[2] Generator mutanţi:
http://pitest.org
http://csharp.academy/mutation-testing/
https://medium.com/analytics-vidhya/unit-testing-in-python-mutation-testing-7a70143180d8
https://pypi.org/project/MutPy/

[3] 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.

[4] Qian Yang, J. Jenny Li, David M. Weiss, A Survey of Coverage-Based Testing Tools. The Computer Journal 52(5): 589-597 (2009)

[5] 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.

Tema 6: Model based Testing with GraphWalker

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

Referințe:

[1] Graphwalker homepage: http://graphwalker.org/

Tema 7: Test Driven Development in C#

- Prezentați această abordare a dezvoltării de software ce combină Test First Development (TFD) cu refactorizarea.
- Conceptele prezentate vor fi ilustrate pe exemple proprii (create de echipă).

Referințe:

[1] https://docs.microsoft.com/en-us/visualstudio/test/quick-start-test-driven-development-with-test-explorer?view=vs-2019
[2] https://www.freecodecamp.org/news/tdd-explanation-hands-on-practice-with-c-a0124338be44/
[3] https://www.c-sharpcorner.com/article/test-driven-development-tdd-part-one/

Tema 8: Unit Testing with C#

- Prezentați facilitățile unui tool pentru testarea unitară în C#.
- Ilustrați strategiile de generare de teste (prezentate în referința [4]) pe exemple proprii (create de echipă).

Referințe:

[1] https://docs.microsoft.com/en-us/visualstudio/test/unit-test-your-code?view=vs-2019
[2] MSTest:
https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-mstest
https://docs.microsoft.com/en-us/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code?view=vs-2019
NUnit: https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-nunit
[3] Generator mutanți: http://csharp.academy/mutation-testing/
[4] Testare functionala, testare structurala, analiza mutanților, note de curs.

Tema 9: Unit Testing with Python

- Prezentați facilitățile unui tool pentru testarea unitară în Python.
- Ilustrați strategiile de generare de teste (prezentate în referința [4]) pe exemple proprii (create de echipă).

Referințe:

[1] PyUnit:
https://realpython.com/python-testing/
https://docs.python-guide.org/writing/tests/
https://www.freecodecamp.org/news/an-introduction-to-testing-in-python/
[2] Generator mutanți: https://pypi.org/project/MutPy/
[3] Testare functionala, testare structurala, analiza mutanților, note de curs.

Tema 10: Automated Testing in Web Applications

- Prezentați facilitățile unui tool pentru testarea automată a paginilor web.
- Ilustrați strategiile de generare de teste pe exemple proprii (create de echipă).

Referințe:

[1] https://www.guru99.com/top-20-web-testing-tools.html
[2] https://www.softwaretestinghelp.com/most-popular-web-application-testing-tools/
[3] https://medium.com/@briananderson2209/best-automation-testing-tools-for-2018-top-10-reviews-8a4a19f664d2

Tema 11: Automated Testing in Mobile Applications

- Prezentați facilitățile unui tool pentru testarea automată a aplicațiilor mobile.
- Ilustrați strategiile de generare de teste pe exemple proprii (create de echipă).

Referințe:

[1] https://www.guru99.com/mobile-testing-tools.html
[2] https://www.softwaretestinghelp.com/5-best-automation-tools-for-testing-android-applications/
[3] https://medium.com/intuz/top-10-automated-testing-tools-for-mobile-apps-8d9380e1757f

Tema 12: Mutation Testing of Mobile Applications

- Prezentați tool-urile de analiză a mutanților pentru aplicații mobile si evidențiați utilitatea operatorilor de mutație pentru evaluarea testelor şi detectarea erorilor pe baza unor exemple de cod proprii (create de echipă).

Referințe:

[1] Lin Deng, Nariman Mirzaei, Paul Ammann, Jeff Offutt, Towards mutation analysis of Android apps. ICST Workshops 2015: 1-10
[2] Lin Deng, Jeff Offutt, Paul Ammann, Nariman Mirzaei, Mutation operators for testing Android apps. Information & Software Technology 81: 154-168 (2017)
[3] Diego Seco Naveiras, Mutation Testing Techniques for Mobile Applications, PhD thesis, 2021
[4] Automated mutation testing for Swift inspired by Stryker, PITest, and Mull, https://github.com/muter-mutation-testing/muter
[5] Eapen, Naived George, et al., Security aspects for mutation testing in mobile applications. Cyber Intelligence and Information Retrieval: Proceedings of CIIR 2021. Springer Singapore, 2022.

Tema 13: Integration Testing

- Prezentați tipurile de teste de integrare pe care le utilizează în mod regulat dezvoltatorii.
- Ilustrați strategiile de generare de teste pe exemple proprii (create de echipă).

Referințe:

[1] https://www.guru99.com/integration-testing.html
[2] https://www.softwaretestinghelp.com/what-is-integration-testing/

Tema 14: Model based Testing with Rodin Platform

- Prezentați facilitățile platformei Rodin pentru modelarea și verificarea consistenței modelelor discrete și hibride.
- Să se prezinte modul de verificare al unor proprietăți de safety cu ajutorul model checker-ului ProB pe baza unui exemplu dat de sistem discret precum și al unuia hibrid. Să se comenteze rezultatele.

Referințe:

[1] Rodin homepage: https://www3.hhu.de/stups/handbook/rodin/current/html/
[2] Michael Butler, Using Event-B Refinement to Verify a Control Strategy, 2009.
[3] Michael J. Butler, Jean-Raymond Abrial, Richard Banach, Modelling and Refining Hybrid Systems in Event-B and Rodin. From Action Systems to Distributed Systems 2016: 29-42.
[4] Theory Plug-in user manual, Theory Plug-in for Rodin 3.x

Tema 15: Model based Testing with Rodin Platform

- Prezentați facilitățile, dar și neajunsurile platformei Rodin pentru modelarea și verificarea consistenței modelelor hibride și necesitatea cosimulării.
- Să se prezinte modul de validare al unui sistem hibrid pe baza unui exemplu dat cu ajutorul plug-in-ului Multisim. Să se comenteze rezultatele.

Referințe:

[1] Rodin homepage: https://www3.hhu.de/stups/handbook/rodin/current/html/
[2] Vitaly Savicks, Integrating formal verification and simulation of hybrid systems, 2014. Capitolele 3, 4.
[3] Hoang, Thai Son, et al., Validating the requirements and design of a hemodialysis machine using iUML-B, BMotion studio, and co-simulation. International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z. Springer, Cham, 2016.
[4] Multisim Plug-in homepage: https://github.com/snursmumrik/multisim
[5] OpenModelica homepage: https://openmodelica.org