EXPANDED COURSE DESCRIPTION
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
Lassonde School of Engineering
Electrical Engineering Computer Science
LE / EECS 3311 3.0 SECTION Z
SOFTWARE DESIGN
FALL 2019 / WINTER 2020

Last Modified Date: 07/18/2019

COURSE CALENDAR DESCRIPTION

A study of design methods and their use in the correct implementation, maintenance and evolution of software systems. Topics include design, implementation, testing, documentation needs and standards, support tools. Students design and implement components of a software system. Prerequisites: cumulative GPA of 4.50 or better over all major EECS courses (without second digit "5"); LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS 2011 3.00; SC/MATH 1090 3.00; LE/EECS 2031 3.00 or LE/EECS 2032 4.00. Course Credit Exclusions: LE/CSE 3311 3.00

INSTRUCTOR(S)

<table>
<thead>
<tr>
<th>Name</th>
<th>Section / Format / Term</th>
<th>Contact Email</th>
<th>Contact Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ostroff, Jonathan</td>
<td>Sec. Z / LECT / W</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ADDITIONAL INFORMATION

Software designers are experts at developing software products that are correct, robust, efficient and maintainable. Correctness is the ability of software products to perform according to specification. Robustness is the ability of a software system to react appropriately to abnormal conditions. Software is maintainable if it is well-designed according to the principles of abstraction, modularity, and information hiding. At the end of the course, students will be able to:

1. **Specification:** Describe software specifications via Design by Contract, including the use of preconditions, postconditions, class invariants, loop variants and invariants
2. **Construction:** Implement specifications with designs that are correct, efficient and maintainable.
3. **Testing:** Develop systematic approaches to organizing, writing, testing and debugging software.
4. **Analysis:** Develop insight into the process of moving from an ambiguous problem statement to a well-designed solution.
5. **Architecture:** Design software using appropriate abstractions, modularity, information hiding, and design patterns.
6. **Tools:** Develop facility in the use of an IDE for editing, organizing, writing, debugging, testing and documenting code including the use of BON/UML diagrams for documenting designs. Also the ability to deploy the software in an executable form.
7. **Documentation:** Develop the ability to write precise and concise software documentation that also describes the design decisions and why they were made.

ACADEMIC INTEGRITY LINKS

- Senate Policy on Academic Honesty - http://secretariat-policies.info.yorku.ca/policies/academic-honesty-senate-policy-on/
- Academic Integrity - http://lassonde.yorku.ca/academic-integrity

STUDENT LINKS

- Student Rights and Responsibilities - http://oscr.students.uit.yorku.ca/student-conduct
LAND ACKNOWLEDGEMENT

We acknowledge our presence on the traditional territory of many Indigenous Nations. The area known as Tkaronto has been care taken by the Anishinabek Nation, the Haudenosaunee Confederacy, the Huron-Wendat, and the Métis. It is now home to many Indigenous Peoples. We acknowledge the current treaty holders, the Mississaugas of the New Credit First Nation. This territory is subject of the Dish With One Spoon Wampum Belt Covenant, an agreement to peaceably share and care for the Great Lakes region.

The Indigenous Framework for York University: A Guide to Action can be found here: http://indigenous.info.yorku.ca/

Meaning of a land acknowledgement: http://healthydebate.ca/opinions/indigenous-land-acknowledgements

Many courses utilize Moodle, York University's course website system. If your course is using Moodle, click here to access it.

Moodle @ York University