Chen-Wei Wang

Curriculum Vitae

Last Updated: Mar 21, 2024

Roadmap

Personal

Name	Chen-Wei (Jackie) Wang
Rank	Associate Professor, Teaching Stream (since July 2022)
Department	Electrical Engineering and Computer Science
Faculty	Lassonde School of Engineering
Institution	York University
Office	LAS 2043, 4700 Keele Street, Toronto, Canada M3J 1P3
Virtual Office	https://yorku.zoom.us/my/jackie.loves.oxford
E-Mails	jackie@eecs.yorku.ca wangcw@yorku.ca
Professional Licence	Eng. L. (Engineering Licensee), APEGS
Field of practice	Computer and Software Engineering: Teaching and Research
Home Page	https://www.eecs.yorku.ca/~jackie
Teaching Gallery	https://www.eecs.yorku.ca/~jackie/teaching/iPad
Lectures Site	https://www.eecs.yorku.ca/~jackie/teaching/lectures
Tutorials Site	https://www.eecs.yorku.ca/~jackie/teaching/tutorials
Teaching Channel	https://www.youtube.com/user/jackiechenweiwang

EDUCATION

Doctor of Philosophy, Computer Science — October 2006 — June 2012

University of Oxford, United Kingdom
Thesis. Model-Driven Development of Information Systems
Advisor. Prof. J. Davies

Bachelor of Arts with Honours, Computer Science — September 2001 — June 2006

York University, Canada
Summa Cum Laude

RECEIVED AWARDS

Educator of the Year — 2020

- *Affiliation.* Lassonde School of Engineering, York University, Canada
- *Nominators.* 135 students/alumni, 19 faculty members (1 MECH & 18 EECS, including Professor J. S. Ostroff and Professor H. Tabassum mentioned in my statement), 5 staff members, and 3 external faculty members: Prof. T. Song, Beijing Institute of Technology; Prof. M. Lee, New Jersey Institute of Technology; and Prof. E. Rahimi, Open University of the Netherlands.
- ***Recognized Accomplishments.*** **1)** dedication to using various technologies to interact with students and create different modes of learning; **2)** using novel ideas in pedagogy to design the sequence of the laboratory components of taught courses; **3)** accessibility to students even in a class of more than 70; and **4)** two high quality ACM quality conference papers on using technology in classrooms.
- *Details.* `https://www.eecs.yorku.ca/~jackie/awards`

Summa Cum Laude — 2006

York University, Canada

Master's Award for Academic Excellence — 2006

Stong College, York University, Canada

Allen S. Berg Award — 2006

Department of Computer Science & Engineering, York University, Canada

NOMINATED AWARDS

President's University Wide Teaching Award (PUWTA) — 2021, 2022

- *Affiliation.* York University, Canada
- *Nominators.* Prof. P. Cribb (Chair, EECS, 2017 – 2020), Prof. P. Liang (Chair of Awards Committee, EECS, 2020 – 2021), Prof. R. Hornsey (Chair, EECS, 2020 – 2022)
- *Faculty Supporter.* Prof. J. S. Ostroff
- ***Accomplishments.*** See "Assistant Professor, Teaching Stream" in Employment History.
- *Result.* Not selected as the recipient (full-time faculty members, less than ten years at York)
- *Remarks from the Senate Committee on Awards.* "The Committee noted that nominees in the Full-Time Faculty category are **all superb** teachers, which made the decision very difficult. The nomination file that Professor Peter Cribb prepared on your behalf was an eloquent tribute to your achievements as a teacher and to the **very positive influence** that you have had on the lives of York students. As a Committee, we spent considerable time reviewing the files which all demonstrate the **high standards** that you have set for teaching at York."

EMPLOYMENT HISTORY

Associate Professor, Teaching Stream	July 2022 — Present
Assistant Professor, Teaching Stream	July 2017 — June 2022

- *Affiliation.* Department of EECS, Lassonde School of Engineering, York University
- ***Accomplishments.*** **1)** demonstrate continuing commitment to quality undergraduate education, by an innovative use of technologies to create **500+** lecture videos and **150+ hours** of tutorial videos; **2)** create an inclusive, engaging, and enriching learning experience via high responsiveness and strong student-teacher relationships; **3)** commit to the continuing professional development and reflections on teaching; and **4)** actively lead and implement initiatives related to the continuing program improvement in the EECS department.

Research Assistant Professor	August 2015 — June 2017

- *Affiliation.* Department of Computer Science, State University of New York (SUNY) Korea
- ***Diversity.*** SUNY Korea was founded in early 2012 as the first American university global campus in South Korea, where the academic curricula and standards are identical to those in Stony Brook University, which is one of the most recognized schools within SUNY.
- *Accomplishments.* **1)** taught undergraduate and graduate courses; and **2)** acted as the director of the Learn Execute Advance Discover (LEAD) Lab to coordinate funded student projects.

Software Engineering Technologist	September 2014 — August 2015

- *Affiliation.* Lassonde School of Engineering, York University
- *Supervisor.* Ulya Yigit, Director of Computing (under guidance of Prof. J. Ostroff)
- *Accomplishments.* **1)** developed an automated assessment toolset for laboratories, assignments, and group projects; **2)** prepared laboratory projects, assignments, and exercises; and **3)** supported projects, assignments, and exercises during laboratory and tutorial hours.
- *Subjects. EECS3311 Software Design*, *EECS3342 System Specification and Refinement*, and *EECS4312 Software Requirements Engineering*

Postdoctoral Research Fellow	July 2012 — August 2015

- *Affiliations.* York University and McMaster Centre for Software Certification
- *Supervisors.* Prof. Alan Wassyng (McMaster University) & Prof. J. Ostroff (York University)
- *Accomplishments.* **1)** worked on the formal verification of nuclear systems, as part of the project *Certification of Safety-Critical Software-Intensive Systems* funded by the Ontario Research Fund for Research Excellence (ORF-RE); **2)** acted as the *Team Lead* of the nuclear domain (February 2014 – August 2015) to interact with industrial partners (OPG and CANDU), to supervise students, and to create industrial *training materials*.
- *Applications.* All courses I taught at York were designed to transfer my expertise on formal methods and safety-critical systems (e.g., sound coding practice, formal system specification).

Teaching Assistant	October 2006 — February 2011

- *Affiliation.* Software Engineering Programme, University of Oxford (*Supervisor.* Prof. J. Davies)
- *Accomplishments.* **1)** helped design contents of practicals (exercises); **2)** demonstrated use of tools; **3)** supervised sessions of practicals; and **4)** led solution sessions.
- *Workload.* 6 one-week graduate-level software engineering modules per year
- *Venues.* University of Oxford and IBM Hursley, UK.
- *Subjects.* 21 instances in total: Object-Oriented Design (1), Object-Oriented Programming (8), Software Product Line (1), Software Testing (6), Concurrency and Distributed Systems (5).

Research Assistant	May 2005 — September 2006

- *Affiliation.* Software Engineering Laboratory, York University (*Supervisor.* Prof. J. Ostroff)
- *Accomplishments.* Developed a compiler to support the formal verification of (a core subset of) the *Eiffel* language (with native support of *Design by Contract*) using Perfect Developer

CONTINUING PROFESSIONAL DEVELOPMENT

Applied Data Science Program	August, 2022

- *Affiliation.* Massachusetts Institute of Technology (MIT) Professional Education
- *Activities.* **1)** reviewed how the Python programming language and statistics are used in data science; **2)** studied subjects of data analysis & visualization, machine learning, pratical data science, deep learning, and recommendation systems through 58 hours of live sessions (lectures by MIT faculty and mentored sessions by industry experts) and weekly hands-on pratical applications; **3)** completed, individually, six quizzes assessing the understanding of lecture contents; and **4)** completed, individually, two mini projects and one month-long capstone project (see: `https://eportfolio.mygreatlearning.com/jackie-wang`).
- *Performance.* Achieved Rank 2 of the class ($\frac{455 \text{ marks}}{460 \text{ marks}} = 98.91\%$) among 150+ participants
- *Course Duration.* 13 weeks (May 7, 2022 – August 7, 2022)
- *Certificate.* See here.

Master Class on Effective Teaching	June, 2021

- *Affiliation.* American Society for Engineering Education (ASEE)
- *Activities.* **1)** defined active learning from a neuroscientific perspective and described the appropriate use of active learning techniques in the classroom; **2)** identified the differences in working memory that can allow instructors to teach more inclusively; **3)** recognized the importance of the default mode network and small breaks in instruction in allowing students to accomplish neural consolidation; **4)** described the use of retrieval practice, spaced repetition, and interleaving to facilitate the development of neural schemas; **5)** recognized the importance of developing sets of neural links through both declarative and procedural pathways to enable both speed and flexibility in learning STEM subjects; **6)** practiced effective teaching using a case study to facilitate collaborative learning; **7)** identified techniques for helping students to avoid procrastination; and **8)** differentiated between biologically primary and biologically secondary materials, and how this affects instruction in STEM.
- *Course Duration.* 3 days (June 21, 2021 – June 23, 2020)
- *Certificate.* See here.

Effective Online Course Design	November, 2020

- *Affiliation.* University of Oxford (Department of Continuing Education)
- *Activities.* **1)** experimented new skills/techniques and shared experience with other professionals from a range of contexts; **2)** gave/received feedback on work, with an emphasis on collaboration and learning together online; and **3)** constructed an online course outline, planed engaging student activities, and considered key issues such as appropriate online student assessment.
- *Course Duration.* 8 weeks (September 14, 2020 – November 8, 2020)
- *Certificate.* See here.

Higher Education Teaching Certificate	September, 2020

- *Affiliation.* Harvard University (Derek Bok Center for Teaching and Learning & HarvardX)

- *Activities.* **1)** explored areas of pedagogy, course and assessment design, professional communication, as well as language and culture of the classroom; **2)** engaged deeply with and reflected on my teaching practices, portfolio, and teaching journey in the higher education (HE) field.
- *Course Duration.* 8 weeks (July 29, 2020 – September 29, 2020)
- *Certificate.* See here.

Teaching Commons, York University

- *Classroom Observation (F17, W20)*
  - *Programs Participated.* Formative Classroom Observation Program (F17), Student Consultants on Teaching at York (SCOTAY) Program (W20)
  - *Activities.* **1)** met with an Educational Developer and a trained student consultant to discuss teaching goals; **2)** conducted reflective dialogues on class visits; and **2)** explored the theories and practice of teaching in higher education.
- *Instructional Skills for Remote Delivery (S21)*
  - *Course Duration.* 2 weeks (June 14, 2021 – June 25, 2021)
  - *Activities.* **1)** engaged in synchronous sessions to deliver a 15-minute lesson in real time; **2)** prepared and shared a second 15-minute lesson, asynchronously; and **3)** received/sent reflective verbal, written, and video feedback from/to the other participants.

PUBLICATION: TEACHING & LEARNING

Refereed Conference Proceedings

1. **Chen-Wei Wang**. *Crafting Technology-Enhanced Educational Videos for Visual Learners.* In *11th Computer Science Education Research Conference (CSERC)*, pp. 24 – 30. ACM, 2022. [Paper]

   Acceptance Rate: 46% (6 out of 13 submissions)

2. **Chen-Wei Wang**. *Creating Tutorial Materials as Lecture Supplements by Integrating Drawing Tablet and Video Capturing/Sharing.* In *8th Computer Science Education Research Conference (CSERC)*, pp. 1 – 8. ACM, 2019. [Paper] [Talk]

   Acceptance Rate: 37.83% (14 out of 37 submissions)

3. **Chen-Wei Wang**. *Integrating Drawing Tablet and Video Capturing/Sharing to Facilitate Student Learning.* In *ACM Global Computing Education Conference (CompEd)*, pp. 150 – 156. ACM, 2019. [Paper] [Talk]

   Acceptance Rate: 33% (33 out of 100 submissions with 317 authors from 25 countries)

4. Jonathan Ostroff and **Chen-Wei Wang**. *Modelling and Testing of Requirements via Executable Abstract State Machines.* In *Model-Driven Requirements Engineering (MoDRE)* (affiliated with *Requirements Engineering*), pp. 1 – 10. IEEE, 2018. [Paper] [Talk]

   Acceptance Rate: 32% (6 long papers out of 19 submissions)

PUBLICATION: SOFTWARE ENGINEERING & FORMAL METHODS

Refereed Journals

1. Linna Pang, **Chen-Wei Wang**, Mark Lawford, and Alan Wassyng. *Formal Verification of Function Blocks Applied to IEC 61131-3.* In *Science of Computer Programming (SCP)*, Volume 113, December 2015, pp. 149 – 190. [Paper]
2. Jim Davies, David Milward, **Chen-Wei Wang**, and James Welch. *Formal Model-Driven Engineering of Critical Information Systems.* In *Science of Computer Programming (SCP)*, Volume 103, June 2015, pp. 88 – 113. [Paper]

Refereed Conference Proceedings

1. **Chen-Wei Wang**, Jonathan Ostroff, and Simon Hudon. *Using Indexed and Synchronous Events to Model and Validate Cyber-Physical Systems.* In *Engineering Safety and Security Systems (ESSS)* (affiliated with *Formal Methods*). Electronic Proceedings of Theoretical Computer Science (EPTCS), Volume 184, pp. 81 – 95, 2015. [Paper]

2. Linna Pang, **Chen-Wei Wang**, Mark Lawford, Alan Wassyng, David Tremaine, Josh Newell, and Vera Chow. *Formal Verification of Real-Time Function Blocks Using PVS.* In *Engineering Safety and Security Systems (ESSS)* (affiliated with *Formal Methods*). Electronic Proceedings of Theoretical Computer Science (EPTCS), Volume 184, pp. 65 – 79, 2015. [Paper]

3. **Chen-Wei Wang**, Jonathan Ostroff, and Simon Hudon. *Precise Documentation and Validation of Requirements.* In *International Workshop on Formal Techniques for Safety-Critical Systems (FTSCS).* Springer's Communications in Computer and Information Science (CCIS), Volume 419, pp. 262 – 279, 2014. [Paper]

4. Jonathan Ostroff, **Chen-Wei Wang**, Simon Hudon, Yang Liu, and Jun Sun. *TTM/PAT: Specifying and Verifying Timed Transition Models.* In *International Workshop on Formal Techniques for Safety-Critical Systems (FTSCS).* Springer's Communications in Computer and Information Science (CCIS), Volume 419, pp. 107 – 124, 2014. [Paper]

5. Linna Pang, **Chen-Wei Wang**, Mark Lawford, and Alan Wassyng. *Formalizing and Verifying Function Blocks using Tabular Expressions and PVS.* In *International Workshop on Formal Techniques for Safety-Critical Systems (FTSCS).* Springer's Communications in Computer and Information Science (CCIS), Volume 419, pp. 125 – 141, 2014. [Paper]

6. **Chen-Wei Wang** and Jim Davies. *Formal Model-Driven Engineering: Generating Data and Behavioural Components.* In *Formal Techniques for Safety-Critical Systems (FTSCS).* Electronic Proceedings of Theoretical Computer Science (EPTCS), Volume 105, pp. 100 – 117, 2013. [Paper]

7. **Chen-Wei Wang**. *Calculating Preconditions for Parallel Workflows.* In *Asia-Pacific Software Engineering Conference (APSEC)*, pp. 499 – 504. IEEE, 2012. [Paper]

8. **Chen-Wei Wang**. *A Formal Approach for the Iterative Design of Behavioural Models.* In *Asia-Pacific Software Engineering Conference (APSEC)*, pp. 505 – 510. IEEE, 2012. [Paper]

9. **Chen-Wei Wang**, Alessandra Cavarra, and Jim Davies. *Formal and Model-Based Testing of Concurrent Workflows.* In *Quality Software (QSIC)*, pp. 252 – 259. IEEE, 2011. [Paper]

10. **Chen-Wei Wang**, Jim Davies, and James Welch. *A guarded workflow language and its formal semantics.* In *Theoretical Aspects of Software Engineering (TASE)*, pp. 25 – 34. IEEE, 2010. [Paper]

11. **Chen-Wei Wang** and Alessandra Cavarra. *Checking model consistency using data-flow testing.* In *Asia-Pacific Software Engineering Conference (APSEC)*, pp. 414 – 421. IEEE, 2009. [Paper]

12. Jonathan Ostroff, **Chen-Wei Wang**, Eric Kerfoot, and Faraz A. Torshizi. *Automated model-based verification of object-oriented code.* In *Verified Software: Theory, Tools, and Experiments (VSTTE)*, pp. 18 – 29, Microsoft Research Report MSR-TR-2006-117, 2006. [Paper]

13. Jonathan Ostroff, **Chen-Wei Wang**, Eric Kerfoot, and Faraz A. Torshizi. *ES-Verify: A Tool for Automated Model-based Verification of Object-Oriented Code.* In Research Tools, *Formal Methods (FM)*, 2006. [Short Paper & Poster]

Technical Reports

1. **Chen-Wei Wang**, Jonathan S. Ostroff, and Simon Hudon. *Using Indexed and Synchronous Events to Model and Validate Cyber-Physical Systems.* Tech Report EECS-2014-03, York University, 2014. [Report]

2. Linna Pang, **Chen-Wei Wang**, Mark Lawford, Alan Wassyng, David Tremaine, Josh Newell, and Vera Chow. *Formal Verification of Real-Time Function Blocks Using PVS.* Technical Report 16, McMaster Centre for Software Certification, McMaster University, 2014. [Report]
3. **Chen-Wei Wang**, Jonathan S. Ostroff, and Simon Hudon. *Precise Documentation and Validation of Requirements.* Tech Report CSE-2013-08, York University, 2013. [Report]
4. Jonathan S. Ostroff, **Chen-Wei Wang**, and Simon Hudon. *TTM/PAT: A Tool for Modelling and Verifying Timed Transition Models.* Tech Report CSE-2013-05, York University, 2013. [Report]
5. Linna Pang, **Chen-Wei Wang**, Mark Lawford, and Alan Wassyng. *Formalizing and Verifying Function Blocks using Tabular Expressions and PVS.* Technical Report 11, McMaster Centre for Software Certification, McMaster University, 2013. [Report]
6. Jonathan Ostroff, **Chen-Wei Wang**, and Simon Hudon. *Precise Documentation of Requirements and Executable Specifications.* Technical Report, Computer Science and Engineering. York University, CSE-2012-03. [Report]

Thesis

1. **Chen-Wei Wang**. *Model-Driven Development of Information Systems.*
   DPhil Thesis, University of Oxford. Oxford University Research Archive. [Thesis]

> *Abstract.* This thesis is aimed at developing reliable information systems through the application of model-driven and formal techniques. These are techniques in which a precise, formal model of system behaviour is exploited as source code. As such a model may be more abstract, and more concise, than source code written in a conventional programming language, it should be easier and more economical to create, to analyse, and to change. The quality of the model of the system can be ensured through certain kinds of formal analysis and fixed accordingly if necessary. Most valuably, the model serves as the basis for the automated generation or configuration of a working system.
>
> This thesis provides four research contributions. The first involves the analysis of a proposed modelling language targeted at the model-driven development of information systems. Logical properties of the language are derived, as are properties of its compiled form—a guarded substitution notation. The second involves the extension of this language, and its semantics, to permit the description of workflows on information systems. Workflows described in this way may be analysed to determine, in advance of execution, the extent to which their concurrent execution may introduce the possibility of deadlock or blocking: a condition that, in this context, is synonymous with a failure to achieve the specified outcome. The third contribution concerns the validation of models written in this language by adapting existing techniques of software testing to the analysis of design models. A methodology is presented for checking model consistency, on the basis of a generated test suite, against the intended requirements. The fourth and final contribution is the presentation of an implementation strategy for the language, targeted at standard, relational databases, and an argument for its correctness, based on a simple, set-theoretic semantics for structure and operations.

TEACHING: COURSE INSTRUCTION

Instances of "[Link to Course]" below may not work in your web browser (to fix, replace %23 by # in your browser's address bar). Alternatively, use Firefox or Chrome browser or Adobe Reader.

Course Instructor — EECS Department, Lassonde, York

1. *EECS 1022 Programming for Mobile Computing*
   - Winter 2018 [Link to Course] [Link to Lectures] [Link to iPad Notes]
   - Winter 2021 [Link to Course] [Link to Lectures] [Link to iPad Notes]
2. *EECS 1021 Object Oriented Programming from Sensors to Actuators*

- Winter 2019 [Link to Course] [Link to Lectures] [Link to iPad Notes]

3. *EECS 2030 Advanced Object Oriented Programming*
   - Fall 2017 [Link to Course] [Link to Lectures] [Link to iPad Notes]
   - Fall 2018 [Link to Course] [Link to Lectures] [Link to iPad Notes]
   - Fall 2019 [Link to Course] [Link to Lectures] [Link to iPad Notes]
   - Fall 2021 [Link to Course] [Link to Lectures] [Link to iPad Notes]
   - Fall 2022 [Link to Course] [Link to Lectures] [Link to iPad Notes]
4. *EECS 2011 Fundamentals of Data Structures*
   - Winter 2022 [Link to Course] [Link to Lectures] [Link to iPad Notes]
   - Winter 2023 [Link to Course] [Link to Lectures] [Link to iPad Notes]
5. *EECS 3311 Software Design*
   - Summer 2015 [Link to Lectures]
   - Fall 2017 [Link to Course] [Link to Lectures] [Link to iPad Notes]
   - Fall 2018 [Link to Course] [Link to Lectures] [Link to iPad Notes]
   - Winter 2019 [Link to Course] [Link to Lectures] [Link to iPad Notes]
   - Fall 2019 [Link to Course] [Link to Lectures] [Link to iPad Notes]
   - Winter 2020 [Link to Course] [Link to Lectures] [Link to iPad Notes]
   - Fall 2020 [Link to Course] [Link to Lectures] [Link to iPad Notes]
6. *EECS 3342 System Specification and Refinement*
   - Winter 2022 [Link to Course] [Link to Lectures] [Link to iPad Notes]
   - Winter 2023 [Link to Course] [Link to Lectures] [Link to iPad Notes]
7. *EECS 4315 Mission Critical Systems*
   - Winter 2023 [Link to Course] [Link to Lectures] [Link to iPad Notes]
8. *EECS 4302 Compilers and Interpreters*
   - Winter 2020 [Link to Course] [Link to Lectures] [Link to iPad Notes]
   - Fall 2022 [Link to Course] [Link to Lectures] [Link to iPad Notes]

Guest Lecturers EECS Department, Lassonde, York

1. **EECS 3311 Software Design**, York University, Canada W2020
   The State and Template Design Patterns 1 Lecture
   [Link to Lecture] [Link to iPad Notes]
2. **EECS 4312 Software Engineering Requirements**, York University, Canada F2019
   Certification of Safety-Critical, Software-Intensive Systems 1 Lecture
   [Link to Lecture] [Link to iPad Notes]
3. **EECS 3311 Software Design**, York University, Canada F2018
   Design by Contract in Java vs. Eiffel 1 Lecture
   [Link to Lecture] [Link to iPad Notes]
4. **EECS 3311 Software Design**, York University, Canada W2018
   The State and Template Design Patterns 1 Lecture
   [Link to Lecture] [Link to iPad Notes]
5. **EECS 2030 Advanced Object Oriented Programming**, York University, Canada F2017
   Unit Testing in Java 2 Lectures
   [Link to Lecture 1] [Link to iPad Notes 1] [Link to Lecture 2] [Link to iPad Notes 2]
6. **EECS 4312 Software Engineering Requirements**, York University, Canada F2014
   Formal Verification of Programmable Logic Controllers 3 Lectures
   1) introduction to programmable logic controllers (PLCs) and programming notations of function blocks (FBs); 2) case studies on formalizing the input-output requirements of FBs and proving the conformance of their implementations; and 3) designed and marked a quiz.

7. **EECS 3311 Software Design**, York University, Canada W2013
   Object-Oriented Design Patterns in Eiffel 4 Lectures
   1) writing contracts using tuples and agents;
   2) composite pattern; 3) decorator pattern; 4) visitor pattern;
   5) adapter pattern; 6) facade pattern; and 7) designed and marked an exam question
8. **EECS 4312 Software Requirements Engineering**, York University, Canada F2012
   Writing Precise Documents for Requirements—the UML approach 4 Lectures
   1) uml diagrams: usecase, class, sequence and state diagrams; 2) model analysis; 3) a train system case study; and 4) designed and marked an assignment

Course Instructor Department of Computer Science, SUNY Korea

Undergraduate

1. *CSE114 Computer Science I (Introduction to Object-Oriented Programming)*
   Fall 2015, Spring 2016, Spring 2017
2. *CSE214 Computer Science II (Data Structures)*
   Fall 2015, Spring 2016, Fall 2016
3. *CSE303 Introduction to the Theory of Computation*
   Spring 2016, Fall 2016, Spring 2017
4. *CSE487 Research in Computer Science*
   Spring 2017

Graduate

1. *CSE547 Discrete Mathematics*
   Fall 2015, Spring 2017
2. *CSE526 Principles of Programming Languages*
   Spring 2016
3. *CSE541 Logic in Computer Science*
   Fall 2016

SCHOLARLY ACTIVITIES

Computing Education & Pedagogy

1. *Using a Drawing Tablet to Craft Educational Videos for Visual Learners*
   Teaching Commons, York University, Canada Aug'21, Oct'21, Nov'21, Jun'22
2. *Creating Technology-Enhanced Lectures and Tutorials for Visual Learners*
   TiF'21 (Teaching in Focus Conference), York University, Canada May'21
3. *Using iPad for Teaching: My Experiences and Best Practices*
   Seminar given to EECS Faculty, York University Canada Aug'20
4. *Creating Tutorial Materials as Lecture Supplements by Integrating Drawing Tablet and Video Capturing/Sharing*
   Computer Science Education Research Conference (CSERC), Larnaca, Cyprus Nov'19
5. *Integrating Drawing Tablet and Video Capturing/Sharing to Facilitate Student Learning*
   ACM Global Computing Education Conference (CompEd), Chengdu, China May'19
6. *Modelling and Testing Requirements via Executable Abstract State Machines*
   Model-Driven Requirements Engineering (MoDRE), Banff, Canada Aug'18

Invited Talks

1. *Using Model Checking and Theorem Proving to Validate and Verify Cyber-Physical Systems*
   National Institute of Advanced Industrial Science and Technology (AIST), Japan 2015
2. *Formal Verification of IEC 61131 Function Block Designs*
   Candu Energy Inc., Mississauga, Canada 2015
3. *A Guarded Workflow Language and its Formal Semantics*
   Academia Sinica, Taipei, Taiwan 2010

Conference Talks

1. *Using Indexed and Synchronous Events to Model and Validate Cyber-Physical Systems*
   Engineering Safety and Security Systems (ESSS), Oslo, Norway 2015
2. Formal Verification of Real-Time Function Blocks Using PVS
   Engineering Safety and Security Systems (ESSS), Oslo, Norway 2015
3. *Formalizing and Verifying Function Blocks using Tabular Expressions and PVS*
   Formal Techniques for Safety-Critical Systems (FTSCS), Queenstown, New Zealand 2013
4. *Precise Documentation and Validation of Requirements*
   Formal Techniques for Safety-Critical Systems (FTSCS), Queenstown, New Zealand 2013
5. *TTM/PAT: Specifying and Verifying Timed Transition Models*
   Formal Techniques for Safety-Critical Systems (FTSCS), Queenstown, New Zealand 2013
6. *Calculating Preconditions for Parallel Workflows*
   Asia-Pacific Software Engineering Conference (APSEC), Hong Kong, China 2012
7. *A Formal Approach for the Iterative Design of Behavioural Models*
   Asia-Pacific Software Engineering Conference (APSEC), Hong Kong, China 2012
8. *Formal Model-Driven Engineering: Generating Data and Behavioural Components*
   Formal Techniques for Safety-Critical Systems (FTSCS), Kyoto, Japan 2012
9. *Formal and Model-Based Testing of Concurrent Workflows*
   International Conference on Quality Software (QSIC), Madrid, Spain 2011
10. *A Guarded Workflow Language and its Formal Semantics*
    Theoretical Aspects of Software Engineering (TASE), Taipei, Taiwan 2010
11. *Checking Model Consistency using Data-Flow Testing*
    Asia-Pacific Software Engineering Conference (APSEC), Penang, Malaysia 2009
12. *Automated Model-Based Verification of Object-Oriented Code*
    Verified Software: Theories, Tools and Experiments (VSTTE), Seattle, USA 2006

SERVICE: CURRICULUM DEVELOPMENT

New Course Development

1. *EECS1015 Introduction to Computer Science*

   Context. This new course was proposed in F19 by the *Programming Course Sequence Committee* (PCS) which I have chaired. It was approved and first offered in F20, completing the first step of the initiative of PCS.

   Cross Reference. See "PCS" in SERVICE: DEPT. OF ELECTRICAL ENGINEERING & COMPUTER SCIENCE (p21).

2. *EECS1022 Introduction to Object Oriented Programming*

   Context. These substantial changes were proposed in F20 and F21 by the *Programming Course Sequence Committee* (PCS) which I have chaired. The revised EECS1022 was approved and first offered in F22, completing the second step of the initiative of PCS.

   Cross Reference. See "PCS" in SERVICE: DEPT. OF ELECTRICAL ENGINEERING & COMPUTER SCIENCE (p21).

3. *EECS4302 Compilers and Interpreters*

   Context. This foundational elective had not been taught since F05. When I taught it in W20, I designed new CLOs with a focus on rigorous software development. Accordingly, I developed new lectures, assignments, tutorial materials, and a final project implementing these CLOs.

Re-Design of Experiential Laboratory Component

Each lab consists of its: **1)** solution; **2)** starter project; **3)** manual; and **4)** tutorial videos (if applicable).

- *EECS3311 Software Design*
  - F20 (3 new labs and final project); W20 (4 new labs and final project);
    F19 (5 new labs); W19 (5 new labs); F18 (5 new labs); and F17 (5 new labs)
  - For W19, W20, and F20, the labs were partly developed by part-time Software Engineering Lab Assistants under the co-supervision of Professor J. S. Ostroff and I. Labs in other terms were designed and developed solely by me.
- *EECS1021 Object Oriented Programming: from Sensors to Actuators*
  - W19 (8 new software labs and 6 new hardware labs)
  - The hardware labs were partly developed by an undergraduate TA under my supervision.
- *EECS1022 Programming for Mobile Computing*
  - W21 (9 new labs) and W18 (6 new labs)
- *EECS4302 Compilers and Interpreters*
  - W20 (3 new assignments and final project)
- *EECS2030 Advanced Object Oriented Programming*
  - F21 (7 new labs); F19 (4 new labs); F18 (4 new labs); and F17 (1 new lab)

  Cross Reference. For EECS1022 and EECS1021, all lab exercises are accompanied with tutorial materials (videos and notes). For EECS2030 and EECS3311, some lab exercises are accompanied with tutorial materials. See "Technology-Enhanced Tutorials for Inclusive Student Learning" in SERVICE: CURRICULUM DEVELOPMENT (p10).

ETF: Eiffel Testing Framework

- Jonathan Ostroff and **Chen-Wei Wang**. *Modelling and Testing of Requirements via Executable Abstract State Machines.* In *Model-Driven Requirements Engineering (MoDRE)* (affiliated with *Requirements Engineering*), pp. 1 – 10. IEEE, 2018.
- *Role.* Main implementer and maintainer (with close collaboration with Prof. J. Ostroff)
- *Applications.* Final projects for EECS3311 (since W15) and EECS4312 (since F14)

Technology-Enhanced Lectures for Inclusive Student Learning

- **Chen-Wei Wang**. *Integrating Drawing Tablet and Video Capturing/Sharing to Facilitate Student Learning.* In *ACM Global Computing Education Conference (CompEd)*, pp. 150 – 156. ACM, May 2019.
- *Applications.* Lectures (**500+ videos** & iPad notes) of my courses since F17 (involving 2,000+ students) have been posted on: `https://www.eecs.yorku.ca/~jackie/teaching/lectures`.
- ***Novelty***. My integration of **1)** a drawing tablet; **2)** design of starter pages (e.g., see a gallery in `https://www.eecs.yorku.ca/~jackie/teaching/iPad`); **3)** systematic and substantial annotations for illustrations; and **4)** open-access sharing of lecture and tutorial videos (listed below) has arguably been a unique practice across departments of Lassonde and York.

- *Cross Reference.* See "Course Instructor (EECS Department, Lassonde, York)" in TEACHING: COURSE INSTRUCTION (p7).

Technology-Enhanced Tutorials for Inclusive Student Learning

- **Chen-Wei Wang**. *Creating Tutorial Materials as Lecture Supplements by Integrating Drawing Tablet and Video Capturing/Sharing.* In *8th Computer Science Education Research Conference (CSERC)*, pp. 1 – 8. ACM, November 2019.
- *Applications.* Experiential laboratory component for all my courses (involving 2,000+ students)
- List of tutorial videos (amounting to **150+ hours**) and notes created using this approach:
- **Introduction to the Rodin Platform for Formal Specifications**
  [Link to Playlist] [Link to iPad Notes]

  Context. EECS2030 (Fall 2021)

  Learning Outcome. Developing, from scratch, an Apple refurbished store application using associations among multiple classes, as well as a set of unit tests using JUnit; Served as a review tutorial, completed in Weeks 1 & 2, covering the assumed OOP basics.

  Duration. **9 hours and 39 minutes (31 videos)**

  Detailed List of Tutorial Videos:

  1. **Developing an Apple Refurbished Store Application in Java**
     [Link to Playlist] [Link to iPad Notes]

     Context. EECS2030 (Fall 2021)

     Learning Outcome. Developing, from scratch, an Apple refurbished store application using associations among multiple classes, as well as a set of unit tests using JUnit; Served as a review tutorial, completed in Weeks 1 & 2, covering the assumed OOP basics.

     Duration. **9 hours and 39 minutes (31 videos)**

     Detailed List of Tutorial Videos:

Part 1: Single Class with Primitive-Typed Attributes

(1.0)	Study Resources	(12:03)	[Link]
(1.1)	Separation of Concerns, Eclipse Work Environment	(20:03)	[Link]
(1.2)	Object Orientation (observe-model-execute), Motivating Problem	(18:28)	[Link]
(1.3)	`Product`: Attributes and Implicit, Default Constructor	(14:02)	[Link]
(1.4)	`Product`: Debugger, Default Values, Addresses, Default Constructor	(18:38)	[Link]
(1.5)	`Product`: Overloaded Constructor, Console Application	(18:47)	[Link]
(1.6)	`Product`: Tracing Object Creations on Debugger vs. Paper	(20:20)	[Link]
(1.7)	`Product`: Generating Getters and Setters	(15:14)	[Link]
(1.8)	`Product`: Accessors `getPrice()` and `toString()`	(18:58)	[Link]
(1.9)	`TestProduct`: Default Constructor – String/int Return Values	(18:43)	[Link]
(1.10)	`TestProduct`: Default Constructor – boolean/double Return Values	(14:39)	[Link]
(1.11)	`TestProduct`: Overloaded Constructor, `assertTrue(str1 == str2)`	(18:37)	[Link]
(1.12)	`TestProduct`: Mutator Method Calls	(16:15)	[Link]
(1.13)	`TestProduct`: Tracing Object Creation and Method Calls	(29:35)	[Link]

Part 2: Inter-Associated Classes with Reference-Typed Attributes

(1.14)	`Entry`: Single-Valued, Reference-Typed Attribute	(14:08)	[Link]
(1.15)	`Entry`: `setProduct` and `toString` (reusing `toString` from Product)	(16:28)	[Link]
(1.16)	`TestEntry`: `assertSame` on object addresses	(16:18)	[Link]
(1.17)	`TestEntry`: `toString` method from `Product` vs. `Entry`	(19:02)	[Link]
(1.18)	`TestEntry`: Changing the Reference of an `Entry`'s Product	(22:31)	[Link]
(1.19)	`TestEntry`: Idea of Reference Aliasing	(09:42)	[Link]
(1.20)	`TestEntry`: Tracing of Reference Aliasing	(44:16)	[Link]

(1.21) `RefurbishedStore`: Multi-Valued, Reference-Typed Attributes (17:12) [Link]
(1.22) `RefurbishedStore`: Adding and Retrieving Entries (18:57) [Link]
(1.23) `TestRefurbishedStore`: Adding Multiple Entries (26:38) [Link]
(1.24) `TestRefurbishedStore`: Tracing Added Entries (debugger) (09:39) [Link]
(1.25) `TestRefurbishedStore`: Tracing Added Entries (paper) (33:39) [Link]
(1.26) `RefurbishedStore`: Modifying an Object via Different Aliases (22:56) [Link]
(1.27) `RefurbishedStore`: Tracing `NullPointerException` in Debugger (11:33) [Link]
(1.28) `RefurbishedStore`: `getSpaceGreyOrPro` – Visual Sketch (15:56) [Link]
(1.29) `RefurbishedStore`: `getSpaceGreyOrPro` – Implementation (10:17) [Link]
(1.30) `RefurbishedStore`: `getSpaceGreyOrPro` – Testing & Debugging (16:36) [Link]

2. Java Programming for Mobile Computing

[Link to Playlist] [Link to iPad Notes]

Context. EECS1022 (Winter 2021), EECS2030 (Fall 2021)

Learning Outcome. Programming from scratch: procedural (assignments, conditionals, loops), object-oriented (classes, attributes, methods, objects), and mobile computing (graphical user interface design, event-driven programming, Android Studio) in Java

Duration. **29 hours and 6 minutes (85 videos)**

Detailed List of Tutorial Videos:

Week 1: Separation of Concerns - Model, Console Apps, Unit Testing

(2.1) Setting Working Environment (Github, Eclipse, Remote Labs) (40:10) [Link]
(2.2) Setting the Java Perspective (15:33) [Link]
(2.3) Console Applications (23:27) [Link]
(2.4) Utility Classes (24:24) [Link]
(2.5) JUnit Tests (21:58) [Link]
(2.6) Separation of concerns (using packages) (13:30) [Link]

Week 2: Use of Debugger and Introduction to Conditionals

(2.7) Arithmetic Sequence - Console Applications (41:23) [Link]
(2.8) Arithmetic Sequence - Utility Methods & JUnit Tests (27:10) [Link]
(2.9) Debugger - Executing Code in Slow Motion (Step Over/Into/Out) (17:00) [Link]
(2.10) Using Debugger in Eclipse (Step Over) (13:17) [Link]
(2.11) Using Debugger in Eclipse (Step Into/Out) (24:01) [Link]
(2.12) Single If-Statement with Overlapping Conditions (11:41) [Link]
(2.13) Multiple If-Statements with Non-Overlapping Conditions (11:52) [Link]

Week 3: Exploration of Logical Operations and Nested Conditionals

(2.14) Setting Up, Reviewing the Grade Example (09:17) [Link]
(2.15) V1 - Single If-Stmt, Overlapping Conditions (18:11) [Link]
(2.16) V2 - Multiple If-Stmts, Disjoint Conditions (10:59) [Link]
(2.17) V3 - Multiple If-Stmts, Disjoint Conditions (18:13) [Link]
(2.18) Specifying Ranges - Conjunction vs. Disjunction (25:22) [Link]
(2.19) Nested If-Statements (30:00) [Link]
(2.20) Helper Method, Using Disjunction, Negation (28:54) [Link]

Week 4: Introducing Loops to Patternize Repetitive Actions

(2.21) Motivating Example of Grade Calculation (with duplicates) (28:24) [Link]
(2.22) Introducing for-Loops in Java (18:03) [Link]
(2.23) Using for-Loops in Console Applications (25:36) [Link]
(2.24) Using while-Loops in Console Applications (16:03) [Link]
(2.25) Utility Method with Loops (a fixed number of iterations) (30:35) [Link]
(2.26) Utility Method with Loops (an indefinite number of iterations) (13:33) [Link]

Week 5: Introducing Arrays – Syntax, Applications, and Tracing

(2.27)	Tracing while-Loops on Paper	(16:57)	[Link]
(2.28)	Tracing while-Loops on Debugger	(15:07)	[Link]
(2.29)	Using while-Loops in Console Applications	(30:22)	[Link]
(2.30)	Arrays: Initializer, Memory, Indexing	(30:14)	[Link]
(2.31)	Arrays: Creating a New Array Object	(12:50)	[Link]
(2.32)	Example: Calculating Intermediate Sums	(19:12)	[Link]
(2.33)	Tracing: Calculating Intermediate Sums	(24:52)	[Link]
	Week 6: Classes, Attributes, Constructors, Accessors/Mutators, Method Calls		
(2.34)	Resources, Creating a Class	(08:41)	[Link]
(2.35)	Class, Attributes, Default Constructor	(19:40)	[Link]
(2.36)	Tracing Object Creations using Debugger	(16:41)	[Link]
(2.37)	Tracing Object Creations on Debugger	(21:49)	[Link]
(2.38)	Tracing Object Creations on Paper	(28:17)	[Link]
(2.39)	Invoking a Helper Constructor	(08:53)	[Link]
(2.40)	Defining & Invoking Accessors and Mutators	(23:20)	[Link]
(2.41)	Syntax: Accessor Methods vs. Mutator Methods	(06:08)	[Link]
(2.42)	Tracing & Visualizing Method Invocations	(23:02)	[Link]
	Week 7: Reference Aliasing and Reference-Typed, Single-Valued Attributes		
(2.43)	Reference Aliasing - Copying an Object Address	(30:51)	[Link]
(2.44)	Reference Aliasing - Swapping Object Addresses	(11:54)	[Link]
(2.45)	Example: A Member's Trainer - Accessors and Mutators	(21:13)	[Link]
(2.46)	Example: A Member's Trainer - Unit Testing	(13:48)	[Link]
(2.47)	Example: A Member's Trainer - Visualization and Tracing	(31:27)	[Link]
(2.48)	Example: A Member's Facilities - Initial Coding and Written Notes	(20:47)	[Link]
	Week 8: Reference Aliasing and Reference-Typed, Multi-Valued Attributes		
(2.49)	Recap, Resources, Setting Up	(10:37)	[Link]
(2.50)	Coding & Testing - Storing Facility Objects	(23:48)	[Link]
(2.51)	Tracing & Visualizing - Additions of Facility Objects	(29:19)	[Link]
(2.52)	Anonymous Objects and Overloaded Methods	(17:10)	[Link]
(2.53)	Coding & Testing - Accessor method `getPaymentDue`	(13:57)	[Link]
(2.54)	Tracing & Visualizing - Accessor Method `getPaymentDue`	(16:57)	[Link]
(2.55)	Coding & Testing - Retrieval and Change of Facility	(24:29)	[Link]
(2.56)	Tracing & Visualizing - Retrieval and Change of Facility	(25:00)	[Link]
	Week 9: Application of OOP in Mobile Computing (Android Studio)		
(2.57)	Greeting App - Creating an Empty Activity	(09:11)	[Link]
(2.58)	Greeting App - GUI Design (activity_main.xml)	(31:43)	[Link]
(2.59)	Greeting App - Attaching Controller Methods to GUI Components	(23:44)	[Link]
(2.60)	Greeting App - Event-Driven Programming (EDP)	(11:56)	[Link]
(2.61)	Counter App - GUI Design (activity_main.xml)	(20:37)	[Link]
(2.62)	Counter App - Attaching Controller Methods to GUI Components	(22:10)	[Link]
(2.63)	Counter App - Tracing EDP and Model-View-Controller (MVC)	(18:35)	[Link]
(2.64)	Counter App - JUnit Tests and Debugger in Android Studio	(09:19)	[Link]
	Week 10: Two-Dimensional Arrays (Part I: Milelage Calculation)		
(2.65)	Problem - Calculating Mileage of Travel	(13:39)	[Link]
(2.66)	Version 1a - Storing Distances Table & Itinerary Cities	(17:32)	[Link]
(2.67)	Version 1a - Explicit Checks on Dep/Dst Cities	(24:10)	[Link]
(2.68)	Version 1b - Implicit Checks on Dep/Dst Cities	(22:48)	[Link]
(2.69)	2D Arrays - Initialization (with Initializer), Lengths, Indexing	(21:54)	[Link]
(2.70)	2D Arrays - Initialization (with new), Indexing, Printing	(20:31)	[Link]

(2.71)	Version 2 - 2D Array Extended with Error Checking	(32:28)	[Link]
	Week 11: Two-Dimensional Arrays (Part II: Utility Methods)		
(2.72)	Example 1: Displaying Elements Row by Row	(21:31)	[Link]
(2.73)	Example 1: Displaying Elements Column by Column	(16:04)	[Link]
(2.74)	Example 2: Overall Average vs. Row Averages	(27:47)	[Link]
(2.75)	Example 3: Returning Max/Min Values	(16:02)	[Link]
(2.76)	Example 4: Returning the Row with Maximum Sum	(17:47)	[Link]
(2.77)	Example 5: Checking the All-Positive Universal Property	(22:03)	[Link]
(2.78)	Example 6: Rectangular Shape?	(17:38)	[Link]
(2.79)	Example 7: Displaying Lower-Left vs. Upper-Left Triangular Areas	(13:38)	[Link]
	Week 12: Java API Case Studies – Building a Birthday Book Application		
(2.80)	Problem - Birthday Book Application	(14:21)	[Link]
(2.81)	BirthdayBook - Method Headers and JUnit Tests	(26:39)	[Link]
(2.82)	BirthdayBook Implementation V1 using ArrayList - Coding	(32:03)	[Link]
(2.83)	BirthdayBook Implementation V1 using ArrayList - Tracing	(28:37)	[Link]
(2.84)	BirthdayBook Implementation V2 using Hashtable - Coding	(17:35)	[Link]
(2.85)	BirthdayBook Implementation V2 using Hashtable - Tracing	(21:34)	[Link]

3. Eiffel: Design Language, Method, and Tool

[Link to Playlist] [Link to iPad Notes]

Context. EECS3311 (Fall 20)

Learning Outcome. Introducing the Eiffel design language, method, and tool, Design by Contract (preconditions, postconditions, and class invariants)

Duration. **9 hours and 15 minutes (23 videos)**

Detailed List of Tutorial Videos:

(3.1)	Unit Testing, Run vs. Run Workbench System	(28:57)	[Link]
(3.2)	Setting Tools Layout, Navigating Library Classes	(20:10)	[Link]
(3.3)	BIRTHDAY: Variable Declarations, Features, Invariant	(20:43)	[Link]
(3.4)	BIRTHDAY: Static Queries, TDD, Boolean Test Case	(28:09)	[Link]
(3.5)	BIRTHDAY: Assertions, Logical Operator Precedence	(25:48)	[Link]
(3.6)	BIRTHDAY: Syntax Overview of Classes and Features	(08:40)	[Link]
(3.7)	BIRTHDAY: Using a Command as a Constructor	(24:06)	[Link]
(3.8)	BIRTHDAY: Setting Breaking Points and Launching Debugger	(15:16)	[Link]
(3.9)	BIRTHDAY: Writing a Precondition Violation Test	(34:45)	[Link]
(3.10)	BIRTHDAY: Writing a Postcondition Test	(30:13)	[Link]
(3.11)	BIRTHDAY: Reference Equality vs. Object Equality	(21:20)	[Link]
(3.12)	BIRTHDAY: Logical Pattern for Invariant	(09:35)	[Link]
(3.13)	Basic Operations of Arrays	(30:58)	[Link]
(3.14)	Basic Operations of Linked Lists	(17:01)	[Link]
(3.15)	Use of `across` as Loop Instructions	(23:46)	[Link]
(3.16)	Use of `across` as Boolean Expressions	(21:28)	[Link]
(3.17)	BIRTHDAY_BOOK: Attributes, Constructor, Void Safety	(27:32)	[Link]
(3.18)	BIRTHDAY_BOOK: Class Invariant	(16:46)	[Link]
(3.19)	BIRTHDAY_BOOK: Command `add` – Debugging Precondition	(30:45)	[Link]
(3.20)	BIRTHDAY_BOOK: Command `add` – Testing Postcondition	(35:16)	[Link]
(3.21)	BIRTHDAY_BOOK: Get Birthday – detachable? [Supplier]	(39:52)	[Link]
(3.22)	BIRTHDAY_BOOK: Get Birthday – detachable? [Client]	(24:49)	[Link]
(3.23)	BIRTHDAY_BOOK: Implement and Specify Query `celebrate`	(18:57)	[Link]

4. Using the ANTLR4 Parser Generator to Develop a Compiler

[Link to Playlist] [Link to iPad Notes]

Context. EECS4302 (Winter 2020)

Learning Outcome. Developing a Compiler by Specifying Tokens and Grammar of the Source Language, Using the ANTLR Tool to Generate a Lexer and a Parser, Designing and Implementing Model Classes, and Transforming ANTLR Parse Trees into Model Objects.

Duration. **4 hours and 55 minutes (7 videos)**

Detailed List of Tutorial Videos:

(4.1)	Installation of `antlr` and `grun`, Setting Up Eclipse Projects	(09:55)	[Link]
(4.2)	Parser Generator, Specifying and Testing Tokens & Grammmars	(45:41)	[Link]
(4.3)	Model Classes, ANTLR Parse Trees, Grammar Labels	(42:27)	[Link]
(4.4)	Implementing Visitors: from Parse Trees to Model Objects	(1:03:08)	[Link]
(4.5)	Console App, Run Configuration, Exporting as Runnable JAR	(48:46)	[Link]
(4.6)	Syntax Errors vs. Semantic Errors, Error Handing by a Listener	(25:38)	[Link]
(4.7)	Adding Semantic Actions, Comparison: Visitor vs. Action	(1:00:10)	[Link]

5. Procedural and Object-Oriented Programming in Java

[Link to Playlist] [Link to iPad Notes]

Context. EECS1021 (Winter 2019), EECS1022 (Winter 2021)

Learning Outcome. Programming from scratch: procedural (assignments, conditionals, loops) and object-oriented (classes, attributes, methods, objects) in Java

Duration. **28 hours and 47 minutes (46 videos)**

Detailed List of Tutorial Videos:

	Lab 1: Simple Console Applications using Primitive Variable Assignments		
(5.1)	Class, main method, print statement, sequential execution	(27:26)	[Link]
(5.2)	Basic numerical literals and operations	(25:16)	[Link]
(5.3)	String literals, concatenating strings and numbers	(33:23)	[Link]
(5.4)	Literals vs. variable, declaring/initializing/re-assigning variables	(32:05)	[Link]
(5.5)	Swapping values of two variables	(33:06)	[Link]
(5.6)	Customizing Java perspective in Eclipse	(10:11)	[Link]
(5.7)	Console Application – Prompt, Read, Process, Output	(47:57)	[Link]
(5.8)	Synchronizing Java projects between personal and lab computers	(9:49)	[Link]
	Lab 2: Simple If-Statements using the Boolean Data Type and Logical Operations		
(5.9)	Closing the latest Github repository, creating a lab Java project	(4:09)	[Link]
(5.10)	Boolean data type and literals, relational operations	(26:37)	[Link]
(5.11)	Logical operations, truth table	(33:21)	[Link]
(5.12)	Negation, double negation	(21:34)	[Link]
(5.13)	Conjunction for including integer interval, *false* by disjunction	(27:09)	[Link]
(5.14)	Disjunction for excluding integer interval, *false* by conjunction	(32:21)	[Link]
(5.15)	If-statement: calculating absolute value	(15:59)	[Link]
(5.16)	If-statement: calculating account withdrawal (`if-else`)	(32:31)	[Link]
(5.17)	If-statement: calculating account withdrawal (`if-elseif-else`)	(27:47)	[Link]
	Lab 3: A Simple Bank Account Application using Nested If-Statements		
(5.18)	A simple bank application (nested if-statements Version 1)	(1:03:08)	[Link]
(5.19)	A simple bank application (nested if-statements Version 2)	(1:03:41)	[Link]
	Lab 4: Syntax and Semantics of for-Loops and while-Loops, Using Breakpoints and Debugger in the Programming IDE to Reveal Defects		
(5.20)	Break points and debugger (swap program, nested if-statements)	(52:28)	[Link]
(5.21)	Introduction to for-loops, flow chart, tracing	(27:34)	[Link]
(5.22)	Console application using for-loops, tracing	(23:33)	[Link]
(5.23)	Introduction to while-loops, flow chart, tracing	(18:57)	[Link]

(5.24) Console application with error handling using while-loops, tracing (43:51) [Link]

Lab 5: Basics of Arrays – Initialization using Loops and Tracing

(5.25) Array initializer, indexing, iteration, exception on bounds (56:45) [Link]
(5.26) Creating a fix-sized array, re-assigning array and elements, tracing (43:00) [Link]
(5.27) Initialing array using a loop, tracing (55:07) [Link]
(5.28) Creating an array with user-input size, tracing (22:32) [Link]

Lab 6: Deciding if Array Elements Universally/Existentially Satisfy Given Properties

(5.29) All numbers positive? (V1 – entire array with accumulation) (44:49) [Link]
(5.30) Some number positive? (V1: entire array with accumulation) (36:01) [Link]
(5.31) All/Some positive? (V2: entire array without accumulation) (26:18) [Link]
(5.32) All/Some positive? (V3/V4: early exit with/out accumulation) (1:02:18) [Link]
(5.33) Array sorted? (V4: early exit without accumulation) (25:31) [Link]

Lab 7: Object Orientation – Classes, Methods, Object Creations, and Method Calls

(5.34) Classes, attributes, constructors, visualizing object creations (1:04:02) [Link]
(5.35) Visualizing object creations using overloaded constructors (41:39) [Link]
(5.36) Use of this keyword in constructors (30:58) [Link]
(5.37) Defining and calling accessor methods, context objects (1:00:03) [Link]
(5.38) Defining and calling mutator methods, context objects (28:25) [Link]
(5.39) Format, expectation, workflows for OO labs and lab test (20:34) [Link]

Lab 8: Understanding and Implementing Associations between Classes

(5.40) `Faculty` class, console tester, aliasing (1:07:20) [Link]
(5.41) `CourseRecrod` class, reference-typed attribute, aliasing (1:23:56) [Link]
(5.42) `Student` class, array-typed attribute, aliasing, `null` reference (1:18:07) [Link]
(5.43) Inter-connected objects with aliasing (44:26) [Link]
(5.44) Alternative version of the `addCoruse` mutator method (30:21) [Link]
(5.45) Defining and using helper methods (44:06) [Link]
(5.46) Adding accessor methods to `CourseRecord` and `Student` classes (28:56) [Link]

6. BMI Calculator: Model, View, Controller

[Link to Playlist] [Link to iPad Notes]

Context. EECS1022 (Winter 2018)

Learning Outcome. Understanding the complete workflow of developing an Android mobile app from scratch

Duration. **5 hours and 52 minutes (21 videos)**

Detailed List of Tutorial Videos:

Part 1.1: Setup (for student using their own computer)

(6.1) Download VirtualBox Tool and Image (5:04) [Link]
(6.2) Setup a Shared Folder between the Host and Virtual Machines (6:51) [Link]

Part 1.2: Setup (for student using a lab machine)

(6.3) Launch Android Studio from the Virtual Machine (1:10) [Link]
(6.4) Create a New Project, Explore the Project Structure (10:59) [Link]
(6.5) Change the Editor Font Face and Size for the Android Project (1:13) [Link]

Part 2: View (as in the Model-View-Controller Pattern)

(6.6) Design of View (as in Mode-View-Controller) (13:57) [Link]
(6.7) Program the GUI Design of View in Android Studio (18:34) [Link]
(6.8) Anatomy of the .xml File for the View (13:30) [Link]
(6.9) Run the app as is with Only the View Implemented (5:01) [Link]

Part 3: Controller (as in the Model-View-Controller Pattern)

(6.10) Attach a Method from Controller to some GUI Component (13:43) [Link]
(6.11) Call Helper Methods to Retrieve User Inputs, Re-Launch the app (19:59) [Link]

(6.12) Convert Inputs to Numbers, Compute the BMI, Re-Launch the app (14:25) [Link]
(6.13) Violation of the Separation of Concern Principle (15:53) [Link]
Part 4: Model (as in the Model-View-Controller Pattern)
(6.14) Tester (with the `main` method) for Test Driven Development (TDD) (28:56) [Link]
(6.15) Object Orientation (entities, classes, attributes, behaviour) (44:18) [Link]
(6.16) Declare a `Person` class and attributes, Crate a Tester for `Person` (19:22) [Link]
(6.17) Define, Test, and Debug the Person Constructor (38:44) [Link]
(6.18) Define, Test, and Debug the getBMI and toString Accessor Methods (38:24) [Link]
(6.19) Define, Test, and Debug the setWeight(double w) Mutator Method (23:50) [Link]
(6.20) Add Code that Manipulates Model Person Objects in the Controller (7:21) [Link]
Part 5: Conclusion
(6.21) Recap: MVC Pattern, OO, TDD (11:50) [Link]

7. Event-Driven Controller vs. Object-Oriented Model

[Link to Playlist] [Link to iPad Notes]

Context. EECS1022 (Winter 2018)

Learning Outcome. Understanding the separation of a controller (listening to users' requests) and a model (specifying the business logic)

Duration. **1 hour and 33 minutes (5 videos)**

Detailed List of Tutorial Videos:

(7.1) Setting Up the Project (5:06) [Link]
(7.2) GUI Design for the View (4:34) [Link]
(7.3) Attaching Controller Methods to GUI Buttons (7:56) [Link]
(7.4) Counter Model: Test-Driven Development (TDD) and Debugger (25:03) [Link]
(7.5) Connecting Controller and Model (Multiple- vs. Shared-Object) (51:18) [Link]

8. Object-Oriented Programming: Reference-Typed Attributes

[Link to Playlist] [Link to iPad Notes]

Context. EECS1022 (Winter 2018)

Learning Outcome. Understanding the distinction between attributes of primitive types and of reference types, Consequence of reference aliasing at runtime

Duration. **48 minutes (2 videos)**

Detailed List of Tutorial Videos:

(8.1) Assignments of Reference Variables, Aliasing (40:15) [Link]
(8.2) Attributes with Reference Types, Mutator Methods (55:57) [Link]

9. Object-Oriented Programming: Array-Typed Attributes

[Link to Playlist] [Link to iPad Notes]

Context. EECS1022 (Winter 2018)

Learning Outcome. Understanding multi-valued class associations at compile time and object aliasing at runtime

Duration. **6 hours and 53 minutes (16 videos)**

Detailed List of Tutorial Videos:

(9.1) Creating a No-Activity Project (2:34) [Link]
(9.2) Defining/Testing the `CourseRecord` Class (31:17) [Link]
(9.3) `Student` – Maintaining an Array of CourseRecord Objects (55:56) [Link]
(9.4) `Student` – Demystifying the Dot Notation (14:12) [Link]
(9.5) `Student` – Alternative Mutator for Adding a CourseRecord (22:26) [Link]
(9.6) `Student` – Aliasing of `CourseRecord` Objects (15:36) [Link]
(9.7) `Student` – Accessor getMarks and Mutator setMarks (41:07) [Link]
(9.8) `Student` – Refactoring `getMarks`/`setMarks` using `indexOfCourse` (27:18) [Link]

(9.9) `Student` – Defining Accessor `getGPA` using Helper Method `getGP` (19:56) [Link]
(9.10) `Student` – Error Handing via Attributes `error`/`errorMsg` (23:32) [Link]
(9.11) `StudentManagementSystem` – An Array of `Student` Objects (1:00:26) [Link]
(9.12) `StudentManagementSystem` – the Dot Notation and Aliasing (17:22) [Link]
(9.13) `StudentManagementSystem` – Adding a `CourseRecord` (27:19) [Link]
(9.14) `StudentManagementSystem` – Getting and Setting Marks (20:21) [Link]
(9.15) `StudentManagementSystem` - Getting the GPA at the SMS Level (13:29) [Link]
(9.16) `StudentManagementSystem` – Error Handing via `error`/`errorMsg` (14:40) [Link]

10. Developing a Birthday Book Application in Java

[Link to Playlist] [Link to iPad Notes]

Context. EECS2030 (Fall 2017, Fall 2018, Fall 2019)

Learning Outcome. Developing, from scratch, a birthday book application using associations among multiple classes, as well as a set of unit tests using JUnit

Duration. **4 hours and 23 minutes (22 videos)**

Detailed List of Tutorial Videos:

(10.1) Problem and Setup (5:11) [Link]
(10.2) `Birthday` – attributes, constructor, and accessors (8:12) [Link]
(10.3) `Birthday` – `toString` method, Precondition for constructor (18:38) [Link]
(10.4) `Birthday` – `equals` method (11:18) [Link]
(10.5) `Entry` – attributes, constructor (V1), accessors, `toString` (12:49) [Link]
(10.6) `Entry` – constructor (V2) (9:10) [Link]
(10.7) `Entry` – `equals` method (12:25) [Link]
(10.8) `Entry` – `setName`/`setBirthday`, generated mutator by Eclipse (10:30) [Link]
(10.9) `Entry` – `setBirthday` generated by Eclipse vs. customized (5:15) [Link]
(10.10) `BirthdayBook` – attributes `Entry[] entries`, `int noe`, methods (11:46) [Link]
(10.11) `BirthdayBook` – `toString` method (case of empty book) (5:12) [Link]
(10.12) `BirthdayBook` – `nameExists` (empty book), helper `indexOf(name)` (10:22) [Link]
(10.13) `BirthdayBook` – `getBirthday` (empty book), helper `indexOf(name)` (7:17) [Link]
(10.14) `BirthdayBook` – overloaded method `getReminders` (16:23) [Link]
(10.15) `BirthdayBook` – `removeEntry` (empty book), `getEntries` (8:33) [Link]
(10.16) `BirthdayBook` – overloaded method `addEntry` (16:30) [Link]
(10.17) `BirthdayBook` – `toString` method (non-empty book) (8:15) [Link]
(10.18) `BirthdayBook` – assertions on `nameExists`/`getBirthday`/`Reminders` (8:07) [Link]
(10.19) `BirthdayBook` – `removeEntry` (non-empty book) (24:05) [Link]
(10.20) `BirthdayBook` – from console interactions to JUnit test method (6:55) [Link]
(10.21) `BirthdayBook` – `addEntry` (existing name) (17:24) [Link]
(10.22) `BirthdayBook` – `equals` method (30:34) [Link]

11. Java Collection Library

[Link to Playlist] [Link to iPad Notes]

Context. EECS2030 (Fall 2018, Fall 2019), EECS1022 (Winter 2021)

Learning Outcome. Visualizing the runtime structures and programming with Java collections such as lists, maps, *etc.*

Duration. **1 hours and 21 minutes (2 videos)**

Detailed List of Tutorial Videos:

(11.1) Use of `ArrayList` (33:15) [Link]
(11.2) Use of `HashMap` (48:00) [Link]

12. Implementing the Composite and Visitor Design Patterns

[Link to Playlist]

Context. EECS3311 (Fall 2017, Fall 2018, Winter 2019, Fall 2019, Winter 2020, Fall 2020)

Learning Outcome. Implementing the two closely-related design patterns from scratch and exploring their runtime structure from the debugger

Duration. **1 hour and 50 minutes (6 videos)**

Detailed List of Tutorial Videos:

(12.1)	Setting up the Project	(9:52)	[Link]
(12.2)	Implementing the Composite Pattern	(37:47)	[Link]
(12.3)	Debugging the Composite Pattern	(7:37)	[Link]
(12.4)	Implementing the Visitor Pattern	(27:48)	[Link]
(12.5)	Debugging the Visitor Pattern	(16:43)	[Link]
(12.6)	Open Closed Principle on the Visitor Pattern	(10:18)	[Link]

13. Use of the Eiffel Testing Framework (ETF)

[Link to Playlist]

Context. EECS3311 (Fall 2018, Winter 2019, Fall 2019, Winter 2020, Fall 2020)

Learning Outcome. Learning about the architecture of ETF supporting a separation of concerns: user interface versus business model, acceptance testing vs. unit testing

Duration. **2 hours and 1 minute (7 videos)**

Detailed List of Tutorial Videos:

(13.1)	Setting Environment Variable for the `MATHMODELS` library	(3:45)	[Link]
(13.2)	Generating a starter project from an abstract user interface file	(9:15)	[Link]
(13.3)	GUI mode vs. command-line (interactive or batch) mode	(18:52)	[Link]
(13.4)	Architecture of classes in clusters `user_commands` and `model`	(23:22)	[Link]
(13.5)	Developing model classes and re-running acceptance tests	(31:41)	[Link]
(13.6)	DbC in model classes, error reporting in command classes	(24:44)	[Link]
(13.7)	Acceptance Tests, Expected Outputs, Actual Outputs	(9:55)	[Link]

14. Managing Software Projects Using Github

[Link to Playlist]

Context. EECS2030 (Fall 2018, Fall 2019, Fall 2021), EECS3311 (Fall 2018, Winter 2019, Fall 2019, Winter 2020, Fall 2020), EECS1021 (Winter 2019), EECS1022 (Winter 2021)

Learning Outcome. Setting up a private repository of software projects and understanding the workflow of development (e.g., `clone`, `commit`, `push`, `pull`)

Duration. **1 hour and 20 minutes (7 videos)**

Detailed List of Tutorial Videos:

(14.1)	Applying for an educational account	(6:56)	[Link]
(14.2)	Confirming approval of education account application	(2:56)	[Link]
(14.3)	Installing a desktop IDF for Github	(2:23)	[Link]
(14.4)	A simple work pattern for Github	(23:31)	[Link]
(14.5)	Creating a private repository for Eiffel software	(9:31)	[Link]
(14.6)	Creating a private repository for Java software	(12:58)	[Link]
(14.7)	Synchronizing projects between lab computer and Github repository	(22:29)	[Link]

15. Writing Java Code Based on Given Tests

[Link to Playlist]

Context. EECS2030 (Fall 2018), EECS1021 (Winter 2019)

Learning Outcome. Understanding the importance of developing code that is compilable, test-driven development

Duration. **31 minutes (2 videos)**

Detailed List of Tutorial Videos:

(15.1)	Implementing the Given API w.r.t. Expected Outputs	(42:42)	[Link]

(15.2)	Implementing the Given API w.r.t. Unit Tests	(20:32)	[Link]

16. **Solutions to Practice Test on Arrays/Loops**

[Link to Playlist] [Link to iPad Notes]

Context. EECS1022 (Winter 2018, Winter 2021), EECS1021 (Winter 2019)

Learning Outcome. Given a specified API, program methods which exhibit the correct behaviour, with respect to a set of given input-output tests.

Duration. **4 hours and 8 minutes (12 videos)**

Detailed List of Tutorial Videos:

(16.1)	Arrays and Loops: Basic Syntax and Concepts	(43:19)	[Link]
(16.2)	Implementing `AverageOf` (and review of coercion and cast)	(10:08)	[Link]
(16.3)	Implementing `allMultiplesOf5` and `atLeastOneMultipleOf5`	(44:19)	[Link]
(16.4)	Implementing `secondMaximumOf`	(13:43)	[Link]
(16.5)	Implementing `reverseOf` (one-counter vs. two-counter versions)	(20:43)	[Link]
(16.6)	Implementing `isReverseOfEachOther` (one-count vs. two-counter)	(23:32)	[Link]
(16.7)	Implementing `getArithSeq` (and discussion of special cases)	(11:23)	[Link]
(16.8)	Implementing `isArithSeq`	(16:02)	[Link]
(16.9)	Implementing `getFibSeq`	(13:26)	[Link]
(16.10)	Implementing `isFibSeq`	(25:27)	[Link]
(16.11)	Implementing `numberOfOccurrences` (and review of `==` vs. `equals`)	(7:29)	[Link]
(16.12)	Implementing `getIndices` (using helper `numberOfOccurrences`)	(19:21)	[Link]

SERVICE: LASSONDE SCHOOL OF ENGINEERING

Technology Enhanced & Active Learning (TEAL)

- 2018 – 2021 (**Vice Chair**: April 2020 – June 2020; **Acting Chair**: April 2020 – May 2020; **Chair**: September 2020 – June 2021)
- *Achievements.*
  - Helped review the LEEF proposals
  - Helped review the York AIF applications
  - Led initiatives to: **1)** Review and streamline Learning Management Systems (LMS) to enhance Technology, Teaching & Learning to provide a way to engage and support student success, in alignment with pedagogy; **2)** Facilitate approval process and recommendations on the Lassonde Education Equipment Fund (LEEF) to support teaching & learning, in alignment with pedagogy; and **3)** Establish a process to pilot innovation to enhance student and academic learning and pedagogy.

Lassonde Curriculum & Students (LCS)

- 2020 – 2022 (Science Curriculum Reviewer)
- 2021 – 2022 (**Chair**)

SERVICE: DEPT. OF ELECTRICAL ENGINEERING & COMPUTER SCIENCE

PCS: Programming Course Sequence Committee

- 2019 – 2022 (**Chair**)
- *Initiative.* Evaluate and revise the foundational programming courses, for the CS undergraduate program, in the EECS department: **1)** *EECS1012 Net-Centric Introduction to Computing*; **2)** *EECS1022 Programming for Mobile Computing*; **3)** *EECS2030 Advanced Object Oriented Programming*; and **4)** *EECS2011 Fundamentals of Data Structures.*

- *Accomplishments.* **1)** Proposed EECS1015 (approved in F19 and first offered in F20), focusing on procedural programming to strengthen the technical competence of students, while maintaining their engagement, as an alternative to EECS1012; and **2)** Proposed substantial changes, including course name, re-designed CLOs, and removal of the mobile computing component, to EECS1022 (approved in F21 and first offered in F22), focusing on object-oriented programming to strengthen the programming proficiency of students, via experiential labs.
- *Ongoing Efforts.* Evaluating EECS2030 and EECS2011, as well as proposing changes, as a consequence of the new EECS1015 and revised EECS1022, for approval.
- *References.* Departmental wiki page here; Background video: `https://youtu.be/HKyqPOKBPeM`.

Computer Science Curriculum Committee

- 2018 – 2019
  - In the last meeting on April 9, 2019, the committee decided to propose a new course EECS1015 as an alternative to EECS1012, and I volunteered to take the lead on drafting a proposal on that course (see PCS above).
- 2020 – 2021

File Preparation Committee (FPC)

- 2019 – 2020 (**Chair**)
  - Completed teaching items (e.g., arranged class visits, collected undergraduate student and graduate TA letters) and coordinated research and service materials.

Search Committee

- Summer 2021
  - *Scope.* Two positions in CS Teaching Stream: one in first-year courses and one in theory/security courses
  - Reviewed candiate files and conducted interviews.

Continuous Program Improvement (CPI) Task Force

- 2019 – 2020
  - Redesigning the Continuous Program Improvement (CPI) process for the Engineering program in the EECS department

ACM Programming Committee

- 2017 – 2018
  - Supervised practice sessions in both Fall and Winter semesters and selected EECS representatives

Computer Security Program Committee

- 2018 – 2019
  - Attended monthly meeting for discussion by following the lead of chair

SERVICE: VOLUNTARY

Using a Drawing Tablet to Craft Educational Videos for Visual Learners YU Teaching Commons

- *Role.* Inventor & Instructor
- *Background.* A brand-new, recurring course offerred to instructors across disciplines at York
- *URLs of Course Instances.*
  1) https://www.yorku.ca/teachingcommons/event/using-a-drawing-tablet-to-craft-educational-videos/2021-08-16/
  2) https://www.yorku.ca/teachingcommons/event/using-a-drawing-tablet-to-craft-educational-videos-2/
  3) https://www.yorku.ca/teachingcommons/event/using-a-drawing-tablet-to-craft-educational-videos-3/2021-11-05/
- *Syllabus.* https://www.eecs.yorku.ca/~jackie/teaching/tc/syllabi/syllabus-creating-educational-videos.pdf
- *Referee.* Dr. Brian Nairn (Educational Developer at YU Teaching Commons)
- *Time*: ≈ 60 hours (prep [30h]; 13:00 – 15:00, Aug 16 – 19 [10h], 2021; 10:00 – 12:00, Oct 12 – 15 [10h], 2021; 09:00 – 11:00, Nov 5 – 26 [10h])

Seminar for Faculty: Using iPad for Teaching EECS Department

- Prepared pre-study materials, delivered talk, posted recording
- *Time*: 4 hours (15:30 – 17:00, Aug 28, 2020)

Undergraduate Summer Student Research Conference Lassonde

- Judge of posters
- *Time*: 1.5 hour (13:30 – 15:00, August 13, 2020)

Search Committee for Lab Technologist EECS Department

- Screened applications; interviewed four candidates; and decided on the final candidate
- *Reference*: Ulya Yigit, Director of Computing
- *Time*: ≈ 10 hours (January 2020)

Search Committee for Software Engineering Lab Assistants EECS Department

- Screened applications; designed interview problems & solutions; interviewed candidates; and decided on the final candidates
- *References*: Ulya Yigit, Director of Computing; Prof. J. S. Ostroff
- *Time*: ≈ 30 hours (Summer 2018, Summer 2019, Summer 2020)

Undergraduate Summer Student Research Conference Lassonde

- Judge of posters
- *Time*: 1 hour (9:45 – 12:45, August 15, 2019)

EECS Declaration Day Info Session Lassonde

- Answered questions, provided insight/advice to first-year engineering students

  Time: 1 hour (16:40 – 17:00, January 12, 2021)

Academic Orientation – Faculty Panel Lassonde

- Answered questions, provided insight/advice to new Computer Science and Security students

  Time: 0.5 hour (15:40 – 16:10, September 3, 2019)
- Answered questions, provided insight/advice to new Lassonde students

Time: 0.5 hour (10:50 – 11:20, January 8, 2021)