# Introduction I

**Overview**

- We will discuss the course content, structure and marking scheme
- We will learn about software engineering, **software quality** and **software quality assurance**

# CSCI 3060U
## Software Quality Assurance

- **Instructor:** Dr. Jeremy S. Bradbury
  *Office hours:*
  - Tuesdays 1:00pm-2:00pm (in-person),
  - Fridays 10:00am-11:00am (virtual),
  - otherwise by appointment
- **Teaching Assistants:** Riddhi More, Emon Roy
  *Office hours:*
  - during labs, otherwise by appointment

# CSCI 3060U

## Software Quality Assurance

| | |
|---|---|
| ▪ **Lectures:** | Tues. 2:10pm-3:30pm<br>Thur. 2:10pm-3:30pm |
| ▪ **Laboratories:** | Mon. 12:40pm-2:00pm<br>Tues. 9:40am-11:00am<br>Tues. 11:10am-12:30pm<br>Tues.11:10am-12:30pm<br>Fri. 12:40pm-2:00pm |

Labs start the week of Jan. 20, 2025

# CSCI 3060U

## Software Quality Assurance

- Textbook:
  - No required textbook
  - We will be using online resources

# CSCI 3060U

## Software Quality Assurance

- Aims of the Course:
  - The primary purpose of this course is to build on what has been learned about software engineering in previous courses. The course will provide details on topics aimed at the development of high quality software systems and allow students to practice what they have learned through a group project.

# CSCI 3060U

## Software Quality Assurance

- Topics:
  - ***Introduction*** *(0.5 weeks)*
    - Introduction to Software Engineering
    - Software Quality - what is it, how is it measured, how is it achieved
  - ***Software Process*** *(0.5 weeks)*
    - Software Process Models - plans for achieving and improving software quality
  - ***Agile Methods*** *(1.5 weeks)*
    - XP, Kanban, Scrum

# CSCI 3060U

## Software Quality Assurance

- Topics:
  - ***Software Testing*** *(5 weeks)*
    - Systematic Testing - what is it, levels of testing, designing for test
    - Black Box Testing - functional, input, output, partitioning and gray box testing
    - White Box Testing - coverage, path, decision and mutation testing
    - Continuous Testing - regression, defect testing
    - Exploratory Testing - manual software testing
    - Traditional Test Automation - test maintenance and analysis, harnesses, tracking, tools

# CSCI 3060U

## Software Quality Assurance

- Topics:
  - ***Security Testing and Analysis*** *(0.5 weeks)*
    - Penetration testing, fuzzing, and more.
  - ***Static Analysis*** *(0.5 weeks)*
    - *analysis techniques that do not require executing a program.*
  - ***Code Reviews*** *(0.5 weeks)*
    - *Manual review of source code*

# CSCI 3060U

## Software Quality Assurance

- Topics:
  - ***Software Metrics*** *(1 weeks)*
    - measurement basics, assessment and prediction
  - ***Test Automation using AI*** *(0.5 weeks)*
    - Using machine learning, deep learning and large language models (LLMs) to automate testing tasks

# CSCI 3060U

## Software Quality Assurance

- **Quality Assurance Tools and Case Studies**
  - Throughout the course we will introduce new quality assurance case studies of real software quality practices and real software bugs
  - *For example:*
    - Chaos engineering at Netflix
    - Software testing at IBM
    - Quality assurance in autonomous vehicles
    - Engineering culture at Spotify

# CSCI 3060U

## Software Quality Assurance

- Marking:

| | |
|---|---|
| **Laboratory Course Project** | **50%** |
| **Term Tests (3)** | **40%** |
| **Participation** | **10%** |

*Notes:*

- *In order to pass this course you **must** pass the individual assessments (i.e., the 40% of the mark consisting of the tests).*
- *Peer evaluations of team members will be conducted and will contribute to the final project mark.*

# More Information?

- Course website: http://www.sqrlab.ca/csci3060u/

CSCI 3060U: Software Quality Assurance | Computer Science – Ontario Tech University

**CSCI 3060U: Software Quality Assurance**

Computer Science – Ontario Tech University

**About the Course (Winter 2025)**

According to the 2024-25 Academic Calendar, the course is described as follows:

> **CSCI 3060U Software Quality Assurance.** *Building on previous software design courses, this course concentrates on the rigorous development of high quality software systems. Topics covered in this course include software process, software verification and validation (testing, inspection), software metrics, and software maintenance. A major team project is an important feature of this course. 3 cr, 3 lec,*

ANNOUNCEMENTS

Labs start the week of Jan. 20, 2025

INSTRUCTOR

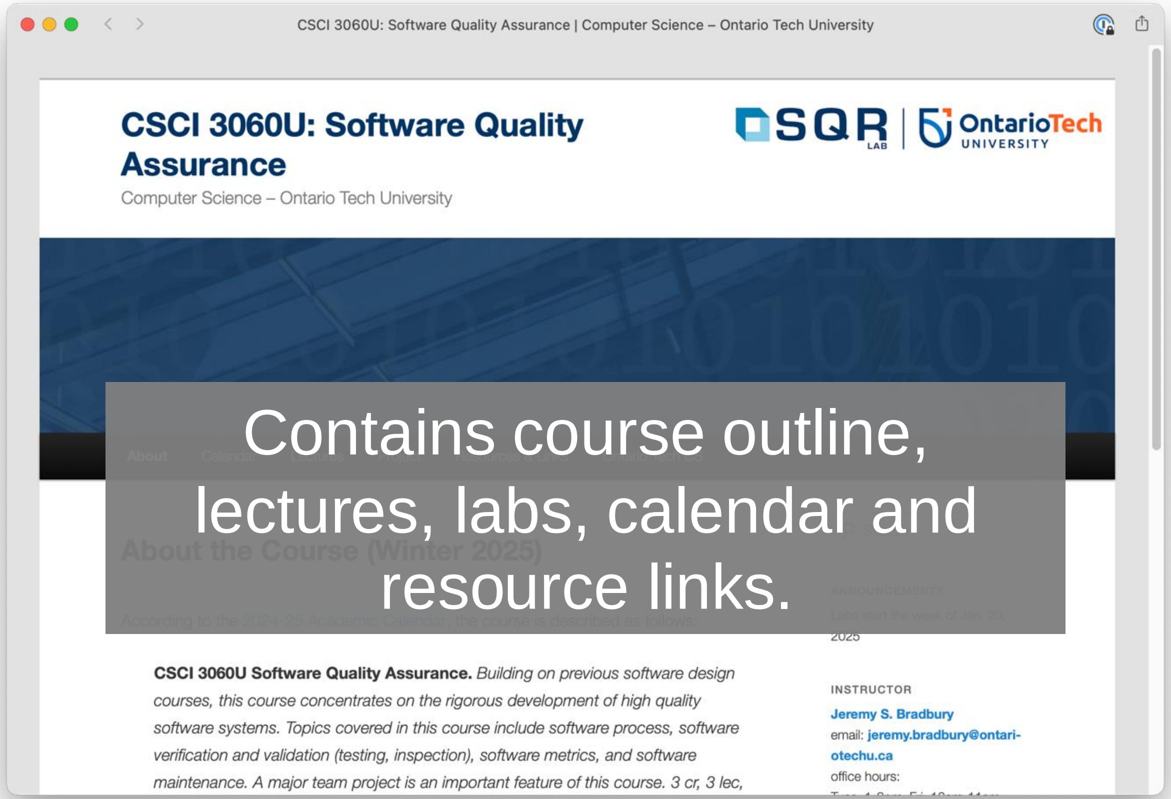
**Jeremy S. Bradbury**

email: **jeremy.bradbury@ontariotechu.ca**

office hours:

# More Information?

- Course website: [http://www.sqrlab.ca/csci3060u/](http://www.sqrlab.ca/csci3060u/)



CSCI 3060U: Software Quality Assurance | Computer Science – Ontario Tech University

**CSCI 3060U: Software Quality Assurance**

Computer Science – Ontario Tech University

Contains course outline, lectures, labs, calendar and resource links.

About the Course (Winter 2025)

According to the 2024-25 Academic Calendar this course is described as follows:

> **CSCI 3060U Software Quality Assurance.** *Building on previous software design courses, this course concentrates on the rigorous development of high quality software systems. Topics covered in this course include software process, software verification and validation (testing, inspection), software metrics, and software maintenance. A major team project is an important feature of this course. 3 cr, 3 lec,*

ANNOUNCEMENTS

Labs start the week of Jan. 20, 2025

INSTRUCTOR

Jeremy S. Bradbury
email: jeremy.bradbury@ontariotechu.ca
office hours:

# Contacting Your Professor/TA

- Slack: https://csci3060u-w25.slack.com

# Contacting Your Professor/TA

- Slack: [https://csci3060u-w25.slack.com](https://csci3060u-w25.slack.com)

Used for discussions, questions & answers and course announcements. If you need to contact your professor or TA – try here first!

# CSCI 3060U

## Software Quality Assurance

- **Course Project**
  - Students will work in small teams
  - As much as possible, the course project will be carried out using agile principles and methods.
  - The project will have 5-6 phases of 1-2 weeks in length. Each phase will have deliverables.

## What is Software Engineering?

*"the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"*

-IEEE

## Quality – What is it?

**Quality →** Not a single idea - many aspects

**Popular View**

- In everyday life, usually thought of as intangible, can be felt or judged, but not weighed or measured
- *"I know it when I see it"* - implies that it cannot be controlled, managed, or quantified
- Often influenced by perception rather than fact

## Quality – What is it?

### Professional View

- In a profession such as software development, there is an ethical imperative to quality
- Quality is not just a marketing and perception issue, it is a moral and legal requirement – we have a professional responsibility associated with the software we create
- Professionals must be able to demonstrate, and to have confidence, that they are using "best practices"

# Quality – What is it?

## Professional View

- In practical terms, therefore, product quality must be measurable in some way
- Product quality is spoken of in terms of
  - conformance to requirements - including timeliness, cost
  - fitness for use - does it actually do the job?
  - freedom from errors and failures - is it reliable and robust?
  - customer satisfaction - are users happy with it?

## So What is Software Quality?

**Software Quality is it…**

- The degree to which a system, component, or process meets specified requirements.

**or**

- The degree to which a system, component or process meets customer or user needs or expectations.

**?**

# So What is Software Quality?

**Software Quality is:** [IEEE Definition]

- The degree to which a system, component, or process meets specified requirements.
  *[based on Philip B. Crosby's definition, 1979]*

- The degree to which a system, component or process meets customer or user needs or expectations.
  *[based on Joseph M. Juran, 1988]*

# So What is Software Quality?

## Software Quality

- Software quality is normally spoken of in terms of several different dimensions often called quality parameters
- These can be split (roughly) into two groups

  Technical Quality Parameters

  - correctness, reliability, capability, performance, maintainability
  - these are open to objective measures and technical solutions (focus of this course)

  User Quality Parameters

  - usability, installability, documentation, availability
  - these often require subjective analysis and nontechnical solutions

# Software Quality

## Technical Quality Parameters

- Correctness - lack of bugs and defects
  - measured in terms of defect rate (# bugs per line of code)
- Reliability - does not fail or crash often
  - measured in terms of failure rate (#failures per hour)
- Capability - does all that is required
  - measured in terms of requirements coverage (% of required operations implemented)
- Maintainability - is easy to change and adapt to new requirements
  - measured in terms of change logs (time and effort required to add a new feature) and impact analysis (#lines affected by a new feature)
- Performance - is fast and small enough
  - measured in terms of speed and space usage (seconds of CPU time, Mb of memory, etc.)

# Software Quality

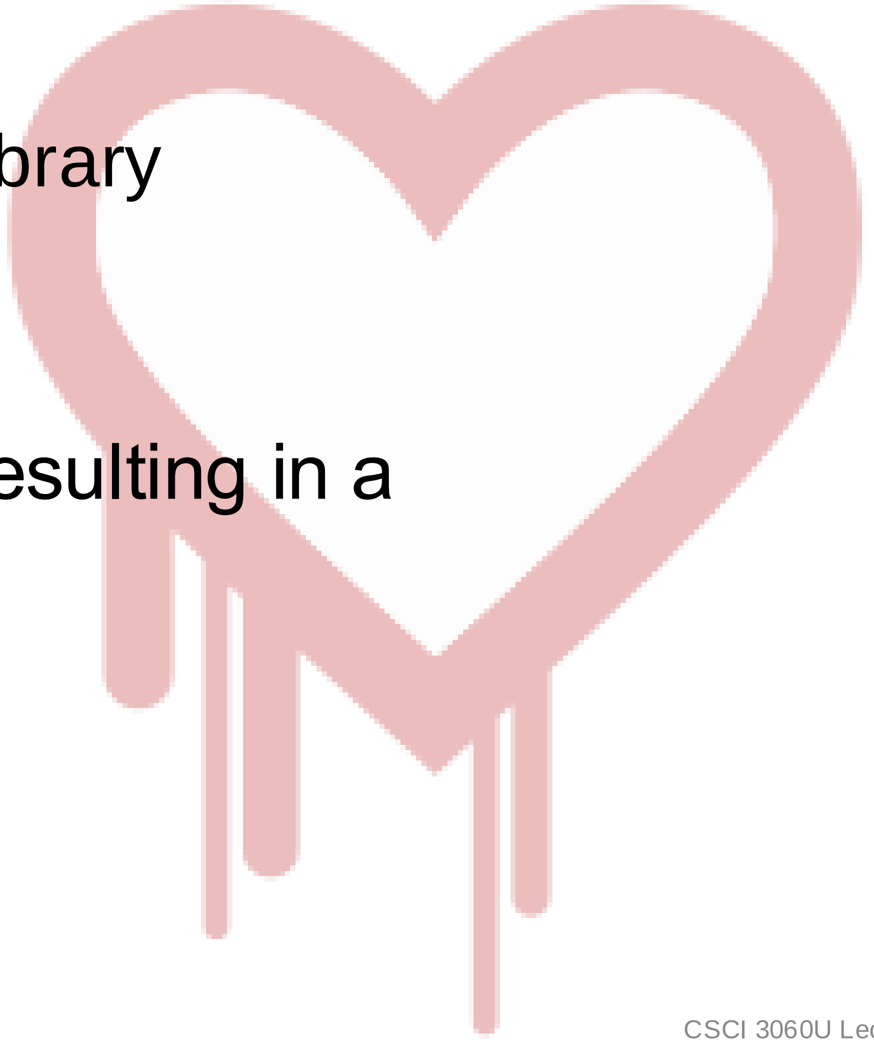
## Technical Quality Parameters

- Correctness - lack of bugs and defects
  - measured in terms of defect rate (# bugs per line of code)
- Reliability - does not fail or crash often
  - measured in terms of failure rate (#failures per hour)
- Capability - does all that is required
  - measured in terms of requirements coverage (% of required operations implemented)
- Maintainability - is easy to change and adapt to new requirements
  - measured in terms of change logs (time and effort required to add a new feature) and impact analysis (#lines affected by a new feature)
- Performance - is fast and small enough
  - measured in terms of speed and space usage (seconds of CPU time, Mb of memory, etc.)

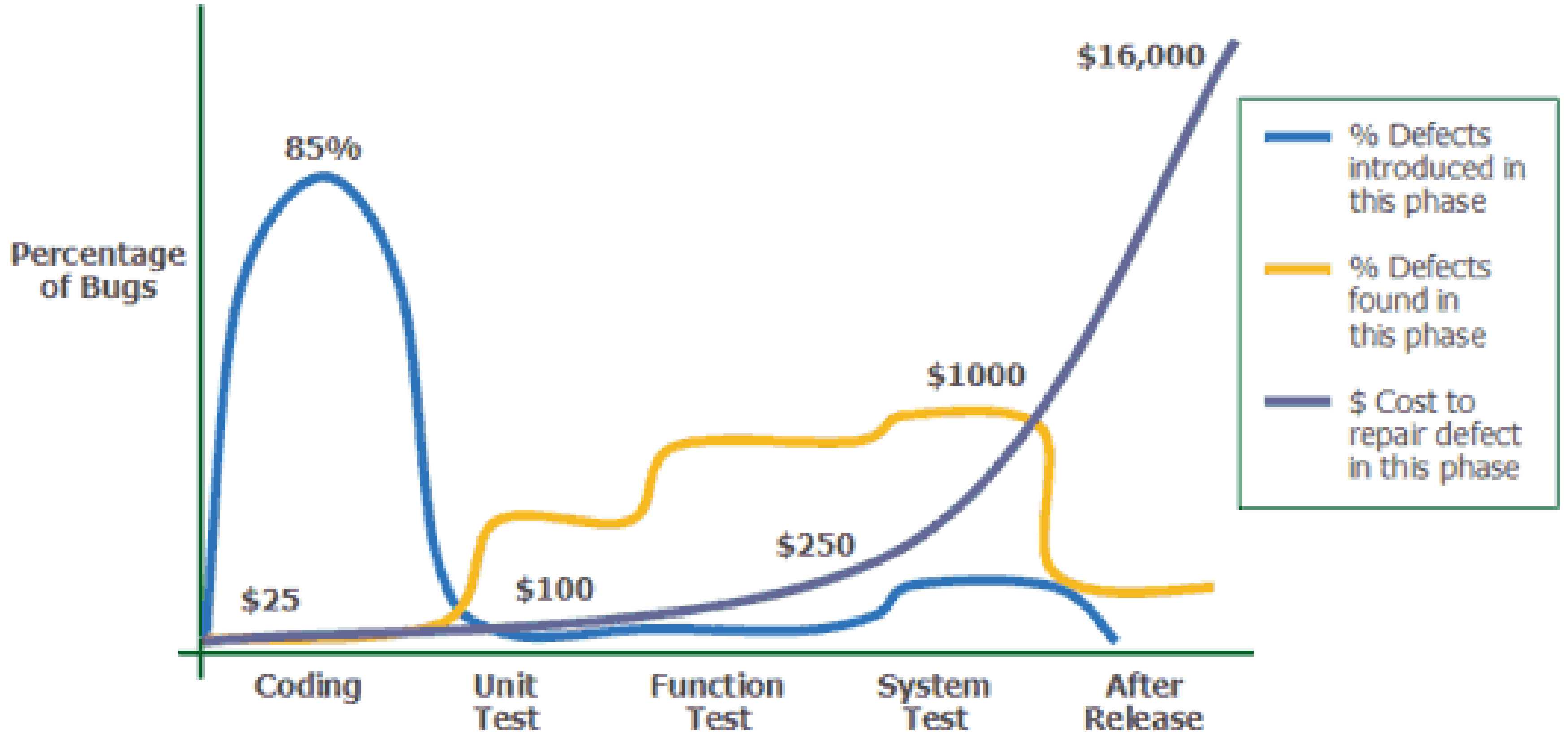
## What causes software bugs?

**Heartbleed**

- A security bug in the cryptography library OpenSSL
- **Cause:** A coding error.
  - Input wasn't validated properly resulting in a buffer over-read



## Why do we care about bugs?



*Source: Applied Software Measurement, Capers Jones, 1996.*

# Software Quality

## User Quality Parameters

- Usability - is sufficiently convenient for the intended users
    - measured in terms of user satisfaction (% of users happy with interface and ease of use)
- Installability - is convenient and fast to install
    - measured in terms of user satisfaction (#install problems reported per installation)
- Documentation - is well documented
    - measured in terms of user satisfaction (% of users happy with documentation)
- Availability - is easy to access and available when needed
    - measured in terms of user satisfaction (% of users reporting access problems)

## Quality Assurance – What is it?

### Achieving Quality

- Product and software quality does not happen by accident, and is not something that can be added on after the fact
- To achieve quality, we must plan for it from the beginning, and continuously monitor it day to day
- This requires discipline
- Methods and disciplines for achieving quality results are the study of Quality Assurance or QA

# Quality Assurance – What is it?

## Three General Principles of QA

1. Know what you are doing
2. Know what you should be doing
3. Know how to measure the difference

# Software Quality Assurance

## QA Principle 1: Know What You Are Doing

- In the context of software quality, this means continuously understanding what it is you are building, how you are building it and what it currently does
- This requires organization, including having a management structure, reporting policies, regular meetings and reviews, frequent test runs, and so on
- We normally address this by following a software process with regular milestones, planning, scheduling, reporting and tracking procedures

# Software Quality Assurance

## QA Principle 2: Know What You Should be Doing

- In the context of software quality, this means having explicit requirements and specifications
- These must be continuously updated and tracked as part of the software development and evolution cycle
- We normally address this by requirements and use-case analysis, explicit acceptance tests with expected results, explicit prototypes, frequent user feedback
- Particular procedures and methods for this are usually part of our software process

# Software Quality Assurance

## QA Principle 3: Know How to Measure the Difference

- In the context of software quality, this means having explicit measures comparing what we are doing to what we should be doing
- Achieved using complementary methods including:
  - Testing - consists of creating explicit inputs or environments to exercise the software, and measuring its success
  - Metrics- consists of instrumenting code or execution to measure a known set of simple properties related to quality

# Achieving Software Quality

## Standards and Maturity Models

- Because of the importance of quality in software production and the relatively bad track record of the past, many standards and maturity models have been developed to enforce quality practice
- Examples inlcude ISO/IEC/IEEE 90003:2018 and the Capability Maturity Model Integration (CMMI)

ICS › 35 › 35.080

**ISO/IEC/IEEE 90003:2018**

**Software engineering — Guidelines for the application of ISO 9001:2015 to computer software**

https://www.iso.org/standard/74348.html

# Introduction I

**Summary**

- We've discussed what quality means, how it applies to software, and the three general principles of quality assurance
- We introduced standards and maturity models

**References**

- Kan, *Metrics and Models in Software Quality Engineering, ch.1*
- *Galin, Software Quality Assurance: From Theory to Implementation, ch 2 & 3*

**Next time**

- We will begin by covering the software development process and see how the software life cycle can affect software quality

# What Causes Software Errors?

1. Faulty definition of requirements
   - Erroneous requirement definitions
   - Absence of important requirements
   - Incomplete requirements
   - Unnecessary requirements included
2. Client-developer communication failures
   - Misunderstanding of client requirements presented in writing, orally, etc.
   - Misunderstanding of client responses to design problems

# What Causes Software Errors?

3. Deliberate deviations from software requirements
   - Reuse of existing software components from previous projects without complete analysis
   - Functionality omitted due to budge or time constraints
   - “Improvements” to software that are not in requirements
4. Logical design errors
   - Errors in interpreting the requirements into a design (e.g. errors in definitions of boundary conditions, algorithms, reactions to illegal operations,...)
5. Coding errors
   - Errors in interpreting the design document, errors related to incorrect use of programming language constructs, etc.

## What Causes Software Errors?

6. Non-compliance with documentation and coding instructions
   - Errors resulting from other team members coordinating with non-  complying member’s code
   - Errors resulting from individuals trying to understand/maintain/test non-complying member’s code
7. Shortcomings of the testing process
   - Incomplete test plan
   - Failure to report all errors/faults resulting from testing
   - Incorrect reporting of errors/faults
   - Incomplete correction of detected errors

## What Causes Software Errors?

8. Procedural errors
   - Incorrect procedures related to user activities that occur in the software
9. Documentation errors
   - Errors in the design documents or code comments
   - Errors in user manuals for software