University of Windsor

Introduction to Software Engineering

Lecture 01: A Gentle Introduction

Agenda

Course Info

Course Instructor Bio

A Gentle Introduction to Software Engineering

Course Description

This course introduces the fundamental concepts, common principles, and general techniques of software engineering. It discusses the main issues involved in the development life-cycle of nontrivial software systems, including process models, feasibility studies, requirements elicitation and definition, rapid prototyping, design methodologies, verification and validation, and software evolution. Students taking this course are required to work on projects, which are designed to go through the major phases of large-scale software system development.

Course Objectives

1. To introduce the students to the principles of the software design, development, and testing.
2. To Introduce the students to different software development lifecycle.
3. Explain various techniques for designing software systems.
4. Expose the students to software design and modeling techniques
5. Create awareness of the common challenges in design, document, evaluate, implement large-scale software systems
6. Practice the design and development of nontrivial software systems

Learning Outcomes

1. Analyze software requirements to identify functional and non-functional software requirements.
2. Select and apply appropriate software process.
3. Design and implement a software system using an agile approach
4. Test and evaluate software quality using a systematic process.
5. Explain common software architecture patterns, software design patterns.
6. Document software requirements specification, design, and test plan.

Course Evaluation

3 Assignments	15% (5% each)
Midterm	15% on Tuesday OCT 16, 2018 during class time
Participation	5% bonus (In-class & Online)
Project	45%
Final Exam	25% on Tuesday DEC 18, 2018 at 7:00 PM

Course Evaluation

- Exams are closed book, and closed notes.
- Assignments: The course has three **individual assignments** each worth 5%. An electronic copy of the assignment should be submitted to the course GA/TA via blackboard.
- Project: Students will execute the project per group. Every group (3 - 5 students per group) will work on a semester-long course project related to software engineering. Each group will design and document and implement a given software system. Another group will evaluate and test the software system, produced by one group. The grading will be done at the group and individual levels.
- Class Participation: Students are expected to participate in class discussions and in-class problem solving.

Course Schedule

Date	Topic	Submission
Week 01 Sep 03 – 07, 2018	Introduction to Software Engineering	
Week 02 Sep 10 – 14, 2018	Software Processes	Submit Project Prospal (3%)
Week 03 Sep 17 – 21, 2018	Software Requirements Engineering	
Week 04 Sep 24 – 28, 2018	Agile Software Development	Submit Assignment 1 (5%)
Week 05 Oct 01 – 05, 2018	System Modelling	Submit Project Part 1 (14 %)
Week 06 Oct 08 – 12, 2018	No Classes (Reading Break)	
Week 07 Oct 15 – 19, 2018	Architectural Design	Midterm Exam
Week 08 Oct 22 – 26, 2018	Design and Implementation	Submit Assignment 2 (5%)
Week 09 Oct 29 – 02, 2018	Software Testing	Submit Project Part 2 (14%)
Week 10 Nov 05 – 09, 2018	Software Evolution	
Week 10 Nov 12 – 16, 2018	Software Security	Submit Assignment 3 (5%)
Week 11 Nov 19 – 23, 2018	Software Reliability	
Week 12 Nov 26 – 30, 2018	Research Directions	Submit Project Part 3 (14%)
Week 13 Dec 03- 07, 2018	Course Wrap-up	

Course Policies

1. Late Assignment: The penalty for late submission of assignments will be 1% for each 24- hour period. No assignment will be accepted later than three days (including weekends) after the deadline.
2. Lecture Attendance: Students are expected to attend all the lectures and tutorials.
3. Coursework Mark Appeals: All marks must be appealed within ten days of the mark being posted.
4. Late Project-deliverable Submission: No late submission for any project-deliverable will be accepted unless prior arrangements have been made with the instructor at least 48 hours before the due date.

Course Instructor and GAs

- Instructor: Sherif Saad - PhD
- Office Hours:
  - Lambton Tower Room 5106,
  - Monday 11:00 – 1:30 PM , Wednesday 11:00 –1:30 PM or by appointment
- GAs:
  - Mr. Farhan Mahmood (MSc Student) - Office Hours (TBD)
  - Ms. Anjali Shah (MSc Student) - Office Hours (TBD)
  - Mr. Havish Kadiyala (MSc Student) - Office Hours (TBD)

Who Am I?

Born and raised In Egypt

I received my PhD in 2015 from University of Victoria, BC Canada

I have 15 years industry experience in software development.

I am certified SCRUM Master

More than 5 years of teaching software engineering courses for university students as a sessional.

What I did

1. Java Developer (2003 - 2006)
2. Senior Software Developer (2006 -2008)
3. Software Architect (2010 - 2013)
4. Software Security Architect (2013 - 2015)
5. Penetration Tester (2009 - 2013)
6. Chief Software Architect (2015 - 2016)
7. Director of Engineers (2016 - 2017)
8. Data Scientist Team lead (2017)
9. WASP Lab Founder and CSO (2017 - present)

Work with and for

ABSOLUTE

VISA

AMERICAN EXPRESS

NU Data Security
A Mastercard Company

ctalyst

lastwall

Plurilock

Work with and for

DEPARTMENT OF DEFENSE
UNITED STATES OF AMERICA

Defence Research and Development Canada

DEFENSE INNOVATION UNIT EXPERIMENTAL

ROYAL CANADIAN MOUNTED POLICE
MAINTIENS LE DROIT
Canada

Booz Allen

VOICE OF THE ARMY

Software Engineering: A Gentle Introduction

The slides are based on “Software Engineering 10th Edition” by Ian Sommerville

Key points

Software Engineering Definition, Activities, and Challenges

Software Types and Properties

Software Engineering Ethics

What is Software?

Software computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.

Software is a set of instructions or programs instructing a computer to do specific tasks.

The software is not limited to program source code. The source code is one of the items a software includes. In addition to the source code, documentation such as design documents, support plan, user guide and troubleshooting documents.

Types Software

There are many categorizations for software, for instance, real-time software, mission-critical software, system software, application software, generic software, custom software, etc.

Different types and categories of software system required different software engineering methods and techniques. Therefore, no one can claim there is one best software engineering technique and method

System Software vs. Application Software

System Software: sometimes called a low-level software is any software used to control the hardware components of the system. Example, the operating system, device driver, and other system utilities.

Application Software: is any software used by the user to accomplish a specific task. An application software requires system software to operate. Examples web browser, media player, word processor, image-editing software.

Generic Software vs Customized

Generic Software: Software systems that are designed and implemented for a generic customer who wishes to use them. Could be commercial or free, open source or proprietary software. Usually suitable for a vast population of users. Identifying and collecting the requirements for generic software is more challenging compared to custom software. Example, Photoshop, JIRA, MS Office, web browser, etc

Customized Software: Software systems that are designed and implemented for specific customers to meet their own needs. Most of the time customized software systems are commercial and proprietary software. Gathering and identifying requirements are less challenging compared to generic software. But on the other hand software testing, verification, and user acceptance is more challenging.

Generic Software vs Customized

Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.

In generic software updates and changes are controlled by the developer not the customer.

In customized software changes and new features are controlled by the customers. The customer usually own and have access to the source code of customized software systems.

Software Engineering: Definition

Software engineering emerged in the late 1960s as a new engineering discipline concerned with all aspects of software production.

Software engineering focuses on concepts, principles, theories, techniques, and tools that can be used for developing high-quality professional software.

Engineering is the creative application of science, mathematical methods, and empirical evidence to the innovation, design, construction, operation, and maintenance of structures, machines, materials, devices, systems, processes, and organizations. (Wikipedia)

Scope of Software Engineering

Software Engineering Vs Computer Science

Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.

Software Engineering Vs System Engineering

System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.

Software Engineer

Programmer vs Software Developer vs Software Engineer

Programmer

Software Developer

Software Engineer

What Does a Software Engineer Do?

Software Engineering Activities & Challenges

What are the fundamental software engineering activities?

Software specification, software design and development, software validation and software evolution.

What are the key challenges facing software engineering?

The increasing complexity, diversity, demands for reduced delivery times and developing trustworthy software.

Software Engineering is Important

By selecting and implementing a successful software engineering process, we will be able to reduce software development complexity, time and cost.

Applying software engineering enables the production of efficient, reliable, and cost-effective software systems.

It is fairly easy to write computer programs without using software engineering methods and techniques. Many companies have drifted into software development as their products and services have evolved. They do not use software engineering methods in their everyday work. Consequently, their software is often more expensive and less reliable than it should be.

Software Process Activities

- Software specification, where customers and engineers define the software that is to be produced and the constraints on its operation.
- Software design and development, where the software is designed and programmed.
- Software validation, where the software is checked to ensure that it is what the customer requires.
- Software evolution, where the software is modified to reflect changing customer and market requirements.

Good Software Properties

The most important attributes of a good software system are maintainability, dependability and security, efficiency, and acceptability.

- Maintainability: Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
- Acceptability: Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

Good Software Properties

- Efficiency: Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency, therefore, includes responsiveness, processing time, memory utilization, etc.
- Dependability & Security: Software dependability includes a range of characteristics including reliability, security, and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.

Software Project Stakeholders

A stakeholder is a person, group, or organization that is actively involved in a project, is affected by its outcome or can influence its outcome.

The term stakeholder is used to refer to any person or group who will be affected by the system, directly or indirectly.

Stakeholders include end-users who interact with the system and everyone else in an organization that may be affected by its installation.

Other system stakeholders may be engineers who are developing or maintaining related systems, business managers, domain experts, and trade union representatives.

Software engineering ethics

Software engineering involves wider responsibilities than simply the application of technical skills.

Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals.

Ethical behaviour is more than simply upholding the law but involves following a set of principles that are morally correct.

Issues of professional responsibility

Confidentiality: Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.

Competence: Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.

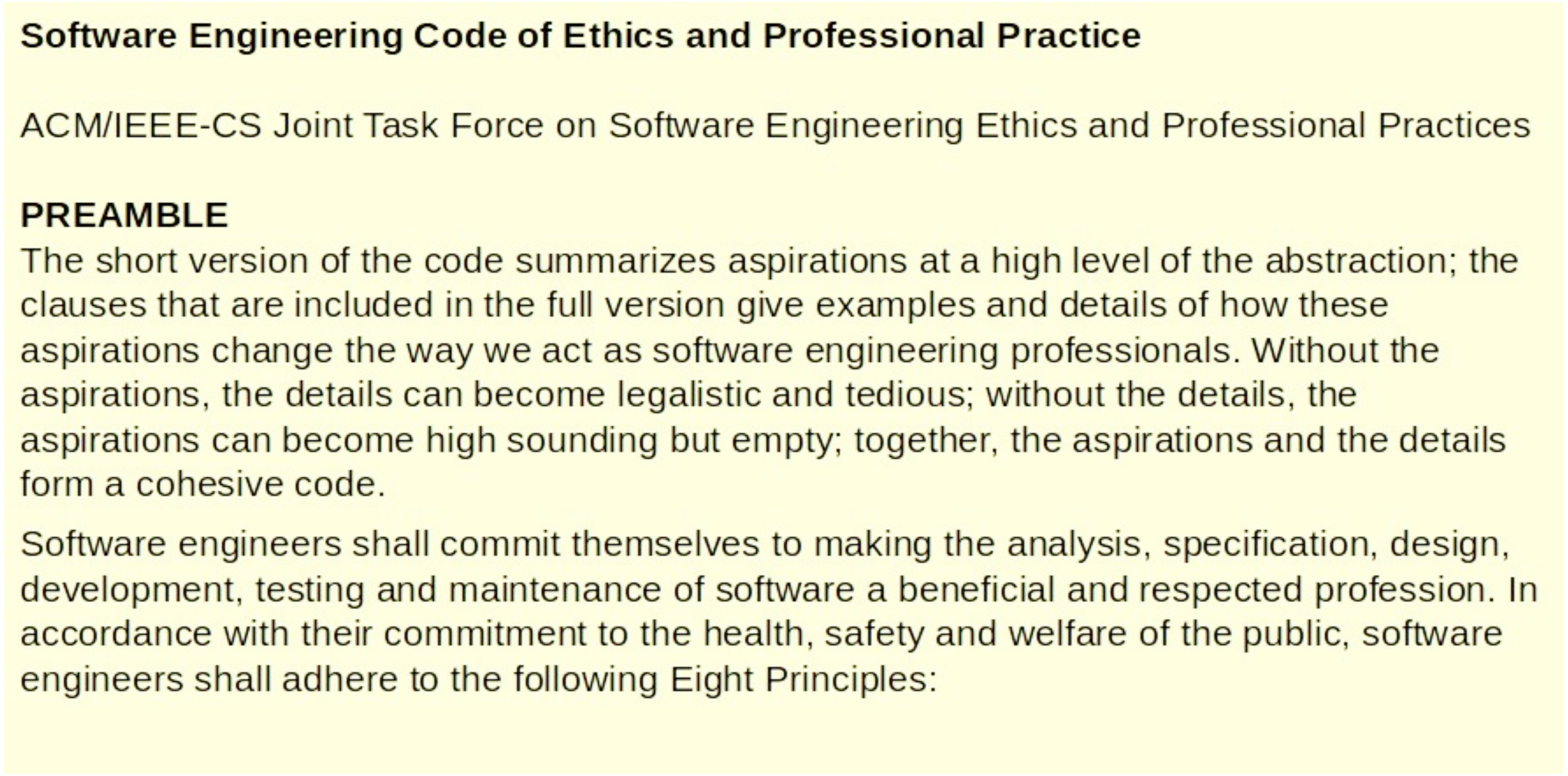
Intellectual property rights: Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.

Issues of professional responsibility

Computer misuse: Software engineers should not use their technical skills to misuse other people’s computers. Computer misuse ranges from relatively trivial (game playing on an employer’s machine, say) to extremely serious (dissemination of viruses).

Example

The ACM/IEEE Code of Ethics



Software Engineering Code of Ethics and Professional Practice

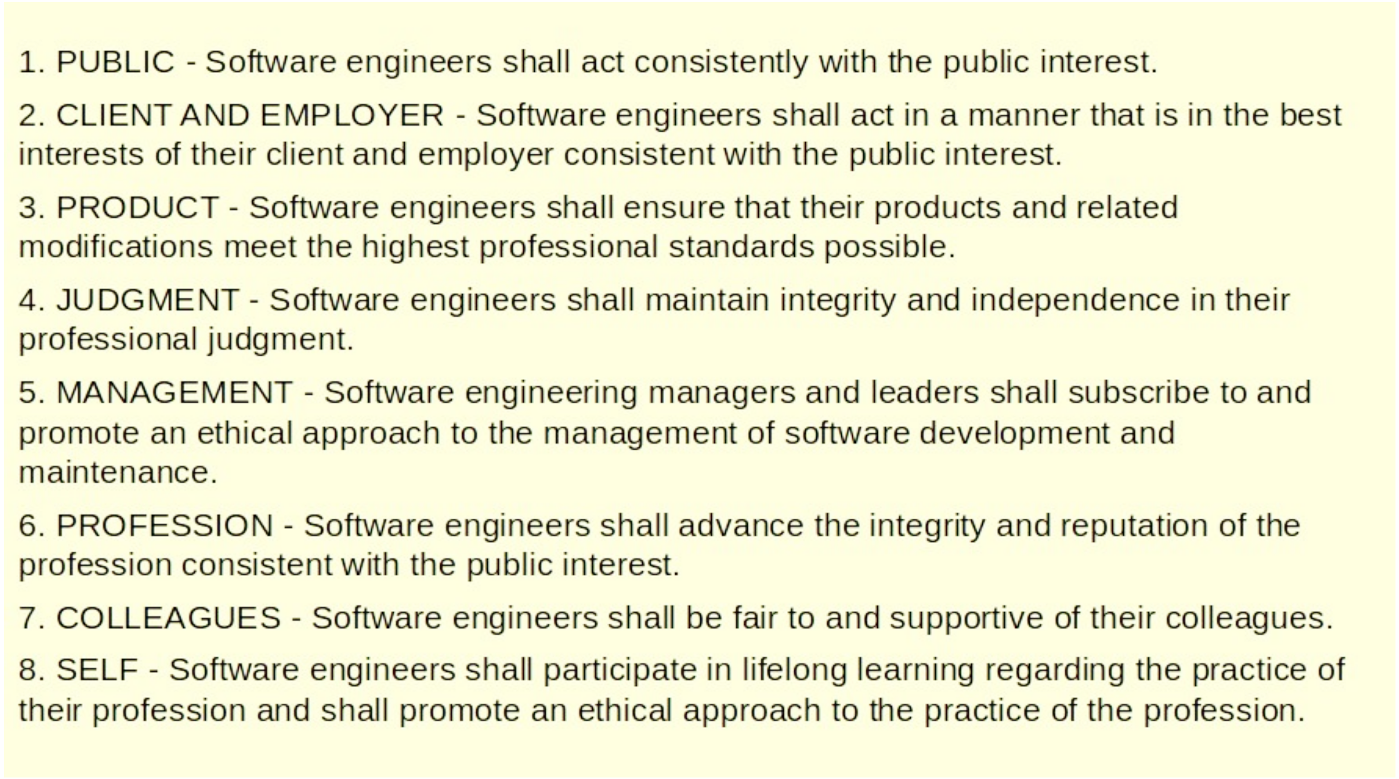
ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

PREAMBLE

The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

The ACM/IEEE Code of Ethics



1. PUBLIC - Software engineers shall act consistently with the public interest.

2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.

3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

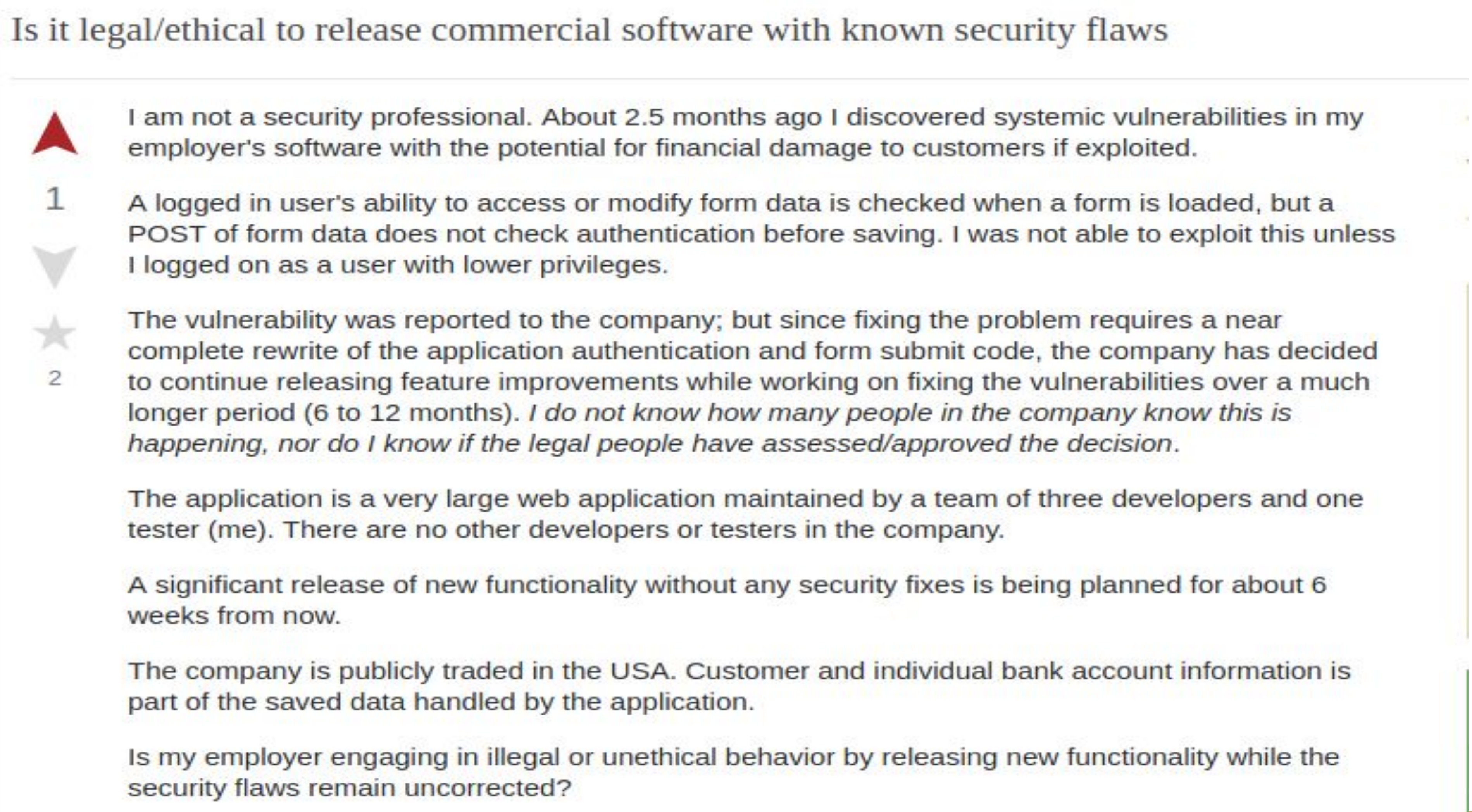
4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.

5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.

8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.



Is it legal/ethical to release commercial software with known security flaws

I am not a security professional. About 2.5 months ago I discovered systemic vulnerabilities in my employer's software with the potential for financial damage to customers if exploited.

A logged in user's ability to access or modify form data is checked when a form is loaded, but a POST of form data does not check authentication before saving. I was not able to exploit this unless I logged on as a user with lower privileges.

The vulnerability was reported to the company; but since fixing the problem requires a near complete rewrite of the application authentication and form submit code, the company has decided to continue releasing feature improvements while working on fixing the vulnerabilities over a much longer period (6 to 12 months). *I do not know how many people in the company know this is happening, nor do I know if the legal people have assessed/approved the decision*.

The application is a very large web application maintained by a team of three developers and one tester (me). There are no other developers or testers in the company.

A significant release of new functionality without any security fixes is being planned for about 6 weeks from now.

The company is publicly traded in the USA. Customer and individual bank account information is part of the saved data handled by the application.

Is my employer engaging in illegal or unethical behavior by releasing new functionality while the security flaws remain uncorrected?

The Challenges of Beginning a Software Engineer

Summary

Software Engineering

Types of Software Systems

Software Engineering Ethics