Syllabus

23 January 2019

> A study of software project management concepts, software cost estimation, quality management, process involvement, overview of analysis and design methods, user interface evaluation, and design. Also considered are dependable systems – software reliability, programming for reliability, reuse, safety-critical systems, verification and validation techniques; object-oriented development; using UML; and software maintenance.

Welcome to CS 164. In this course, we will learn the process behind software development from front to back, by working on a real project through the whole semester. You will be responsible for many parts of the system yourself, but we will discuss the overall design and direction as a class so that we can stay on track and learn from one another.

When: Monday, Wednesday 12–1:50 PM

Where: H-701

Credits: 3

Prerequisites: CS130

Contact Info

Instructor: Prof. Christopher League, Ph.D.

Email: `christopher.league@liu.edu`[1] — please **include "CS164"** in the subject. I have several email addresses, but all messages end up in the same place, so please use only one.

Office hours: Monday, Wednesday 4–4:50 PM and by appointment using `bookme.liucs.net`[2]

Office phone: +1 718 488 1137 (but email is better)

Office location: Pratt 122 (2nd aisle, 2nd desk on left)

Resources

We will use several web resources:

- `liucs.net/cs164s19/`[3] – notes, schedule assignment handouts
- `gitlab.liu.edu`[4] — discussion forum, assignment submission, feedback

[1] christopher.league@liu.edu?subject=CS164

[2] bookme.liucs.net/

[3] liucs.net/cs164s19/

[4] gitlab.liu.edu/

We will also use a designated *virtual machine* image to complete assignments, so that we have a consistent work environment. More details about configuring and operating it will be available in Check-in 1 and Milestone 1.

There is no required textbook, but if you'd like a book to supplement or for reference, here are some suggestions:

- *The Pragmatic Programmer*[5] by Andrew Hunt and David Thomas
- *Debugging Teams: Better Productivity through Collaboration*[6] by Brian W. Fitzpatrick and Ben Collins-Sussman
- *The Effective Engineer*[7] by Edmond Lau

[5] amzn.to/1AkWwHo

[6] amzn.to/1ZlFh5u

[7] www.effectiveengineer.com/book

Requirements

Your grade will be computed based on assignments, exams, quizzes, and participation. There are a total of 1,000 points available, broken down as follows:

- There will be **7 project milestones** scheduled throughout the semester. The exact requirements and expectations for each will be posted to the course web site. Your contribution will be worth **125 points each,** but I will drop the lowest, so that only 6 milestones count, for a total of **750 points. Warning:** the *last* milestone cannot be dropped.
- There will be **7 'check-in' opportunities** scheduled. These vary from week to week, but may involve responding to a survey, taking a brief online quiz, or participating in a discussion forum. Check-ins are worth **25 points each,** but I will drop the lowest two scores so only 5 will count, for a total of **125 points.**
- There is no midterm exam, but there will be a final exam, worth **125 points.**

On the 1,000-point scale, you can expect the following letter grades:

>= 930:	**A**	>= 770:	**C+**
>= 900:	**A–**	>= 730:	**C**
>= 870:	**B+**	>= 680:	**C–**
>= 830:	**B**	>= 600:	**D**
>= 800:	**B–**	else:	**F**

In the end, I may choose to adjust the scale slightly to compensate for assignments or questions that turned out to be trickier than I intended. Such adjustments would never *lower* your grade from what is designated in the above table; if you achieve 930 points, you are guaranteed an **A**.

Policies

It is important to **complete tasks on time,** so you don't fall behind. Missed check-ins will receive a zero, and cannot be made up (but remember, the lowest two are dropped). **Late milestones** are accepted up until finals week, but will be penalized as follows.

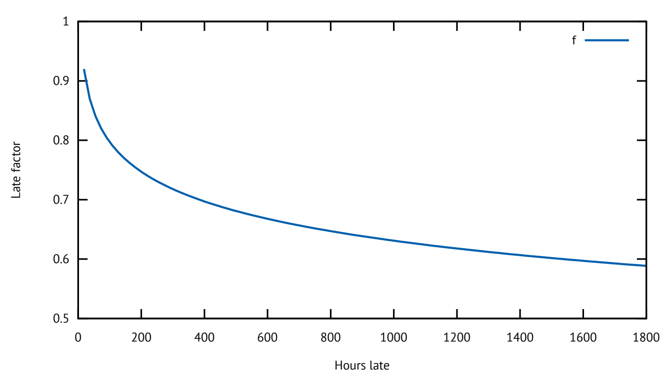
We define a *lateness factor* f as a real number in the range $\{0 \dots 1\}$ that will be multiplied by your earned score to determine a late score. The formula is:

$$f = \min\left(1.0, \frac{18 - \log_2\left(\frac{h}{24}\right)}{20}\right)$$

where the variable h represents the number of hours the submission is late. The table below shows some sample values of the late factor for increasingly late submission times.

weeks late	days late	hours late (h)	late factor (f)
0.01	0.1	2.4	1.000
0.04	0.3	7.2	0.987
0.07	0.5	12.0	0.950
0.14	1.0	24.0	0.900
0.29	2.0	48.0	0.850
0.43	3.0	72.0	0.821
1.00	7.0	168.0	0.760
2.00	14.0	336.0	0.710
4.00	28.0	672.0	0.660
8.00	56.0	1344.0	0.610

The idea is that is that the penalty is somewhat steep initially (from an **A** to a **B+** after just one day) but shallows out over time. It will still be worthwhile to submit a missing assignment, even weeks late.



There will be no extra credit. Students usually ask for extra credit late in the semester after they have already squandered their original opportunities. Be sure to start your work early, so that we can detect and solve any problems before they can impact your grade.

Plagiarism is the use or presentation of ideas, words, or work that is not one's own and that is not common knowledge, without granting credit to the originator. Plagiarism is a practice that is not only unacceptable, but which is to be condemned in the strongest terms possible on the basis of moral, educational and legal grounds. Under University policy, plagiarism may be punishable by a range of penalties from a failing grade in the assignment or course to dismissal from the School of Business, Public Administration and Information Sciences. All students are required to read the handbook on avoiding plagiarism.[8]

[8] `liucs.net/u2`

Cheating includes, but is not limited to the following: falsification of statements or data; listing sources that have not been used; having another individual write your paper or do your assignments; writing a paper or creating work for another student to use without proper attribution; purchase of paper or research work for one's submission as their own work; using written, verbal, or electronic or other sources of aid during an examination (except when expressly permitted by the instructor, depending on the nature of the examination) or knowingly providing such assistance to aid other students.

In a course with programming assignments, it is usually okay to work with and learn from other students to **some** extent, but what you submit in the end needs to be your own. The most reliable way to do that would be to set aside whatever code you created together, and then recreate it from scratch on your own.

Showing up on time to class is extremely important. If you must be absent or more than 5 minutes late, please try to notify me in advance. I will be keeping track of whether you are in class, and when you arrive. A few missed classes will not count against you, but habitual absence will significantly hurt your grade. Additionally, there will be no make-up quizzes. I do not distinguish between "excused" and "unexcused" absence. Unless you miss an **exam** due to a severe medical emergency, I don't need to see a doctor's note. If you do miss an exam, the make-up exam may be somewhat different from the one given in class.

In accordance with Section 504 of the Rehabilitation Act of 1973 and the Americans with Disabilities Act of 1990, including changes made by the Americans with Disabilities Amendments Act of 2008, the Long Island University **does not discriminate against qualified individuals with disabilities.** If you are a student with a documented disability/impairment (psychological, neurological, chronic medical, learning disability, sensory, physical) and require reasonable accommodations, please register with Student Support Services and provide me with an accommodation letter. Visit Sloan Building 1st floor, call 718 488 1044, or visit Student Support Services.[9]

[9] `www.liu.edu/Brooklyn/SSS`

I participate in the **LIU Safe Zone** program. Representatives of the program serve as contacts for individuals on campus with questions or concerns related to sexual orientation and gender identity, whether of self or of a friend or family member. The goal of the program is to promote a safe and free campus for all students. Safe Zone areas can be identified by a sticker with the LIU Safe Zone logo.

The **Family Educational Rights and Privacy Act** (FERPA) gives students control over the disclosure of their educational records. During this course you may have the opportunity to create accounts or register with certain public online services. In these cases, you need not make any personally identifying information public. You may use a pseudonym or online handle, as long as you identify yourself to the instructor.

Goals and objectives

Upon completion of the course, students should be able to…

- demonstrate proficiency in basic algorithms and data structures (1.1, mastery level).
- understand the mathematical and logical foundations of computing (1.2, mastery level).
- master the fundamental facilities of various programming languages and software architectures (2.1, mastery level).
- effectively use tools for software development (2.2, mastery level).
- develop a data modeling design for a proposed database application (3.2, mastery level).
- communicate technical ideas and specifications in writing (4.1, introductory level).
- give an effective oral presentation on some technical subject area (4.2, introductory level).
- exhibit awareness of professional organizations and technical opportunities (5.1, mastery level).
- productively attend seminars and workshops outside of class work (5.2, mastery level).

Schedule

We will examine topics in all the areas of the software development life-cycle:

1. Requirements analysis and project planning
2. Software architecture and system design

3. Implementation tools and techniques
4. Verification and validation
5. Deployment and maintenance

The day-by-day schedule is shown below, including all deadlines. **Topics listed for each date are tentative** though, and subject to change. The schedule is available as `schedule.ics`[10] – copy that link to subscribe or import it into Google Calendar and other systems.

[10] `liucs.net/cs164s19/schedule.ics`

***Wed 23 Jan*: Meeting 1**

Systems Development Lifecycle (SDLC) and software engineering process models

***Mon 28 Jan*: Meeting 2**

Requirements analysis and estimation

***Tue 29 Jan*: Check-in 1 due**

Set up gitlab account and VM.

***Wed 30 Jan*: Meeting 3**

Version control overview

***Sun 3 Feb*: Milestone 1 due**

User stories

***Mon 4 Feb*: Meeting 4**

Testing overview

***Wed 6 Feb*: Meeting 5**

Unit testing tools

***Sun 10 Feb*: Check-in 2 due**

(Skipped)

***Mon 11 Feb*: Meeting 6**

'Assertive' programming

***Wed 13 Feb*: Meeting 7**

Debugging methodology

***Mon 18 Feb*: Milestone 2 due**

Unit testing

***Mon 18 Feb*: No class – Presidents' Day**

***Tue 19 Feb*: Meeting 8**

Coverage analysis

***Wed 20 Feb*: No class – I am out of town**

We will substitute some online content and exercises. Test automation

***Sun 24 Feb*: Check-in 3 due**

Fix Python code that failed tests.

***Mon 25 Feb*: Meeting 10**

Continuous integration

***Wed 27 Feb*: Meeting 11**

Release management

***Sun 3 Mar*: Milestone 3 due**

Coverage tool

***Mon 4 Mar*: Meeting 12**

Stress testing and fuzz testing

***Wed 6 Mar*: Meeting 13**

Mastering your editor and IDE

***Sun 17 Mar*: Check-in 4 due**

Readings on modularity

***Mon 18 Mar*: Meeting 14**

Coupling, cohesion, and modular design

***Wed 20 Mar*: Meeting 15**

Pub-sub and similar models of inter-module communication

***Mon 25 Mar*: Meeting 16**

Model-View-Controller (MVC) and similar models of flexible collaboration

***Wed 27 Mar*: Meeting 17**

Design considerations for our `webgc` project.

***Sun 31 Mar*: Milestone 4 due**

***Sun 31 Mar*: Check-in 5 due**

Skipped

***Mon 1 Apr*: Meeting 18**

Documentation generators, literate programming

***Wed 3 Apr*: Meeting 19**

Doc tests

***Sun 7 Apr*: Milestone 5 due**

***Mon 8 Apr*: Meeting 20**

Dependency injection

***Wed 10 Apr*: Meeting 21**

Mock methods for testing

***Mon 15 Apr*: Meeting 22**

Branching and merging

***Tue 16 Apr*: Check-in 6 due**

***Wed 17 Apr*: Meeting 23**

'Bisect' and other advanced uses of version control

***Sun 21 Apr*: Milestone 6 due**

***Mon 22 Apr*: Meeting 24**

Meta-programming

***Wed 24 Apr*: Meeting 25**

Software verification overview

***Sun 28 Apr*: Check-in 7 due**

Read some introductions about formal methods.

***Mon 29 Apr*: Meeting 26**

Design by Contract (DBC)

***Wed 1 May*: Meeting 27**

Other formal methods

***Mon 6 May*: Meeting 28**

Wildcard/overflow day

***Tue 7 May*: Milestone 7 due**

***Sun 12 May*: Final exam due**