COS 497: Computer Science Capstone 2

Sudarshan S. Chawathe

University of Maine

Spring 2010

This course is the third of a three-course sequence designed to guide students in completing the Capstone project in either an independent study, group project, or field experience format. The focus is on the later stages of project work, including completing the programming tasks, evaluating the implemented systems, documenting all work in a project report, demonstrating the work in action, and making a public oral presentation.

News and Reminders:

- Please read the class newsgroup for timely announcements: `umaine.cs.capstone` on NNTP server `news.cs.umaine.edu`. Web interface to get started: `http://cs.umaine.edu/~chaw/news/`.
- The most recent version of this document may be found at `http://cs.umaine.edu/~chaw/cap2/`.
- Please use the PDF version of this document for printing and reference: `cap2.pdf`

Goals and Learning Objectives

Goals

- Develop the ability to independently explore a topic by discovering, reading, and critiquing prior work.
- Gain experience in contributing to the body of knowledge.
- Gain experience in conducting and documenting experimental studies of programs.
- Improve our programming skills, with attention to software engineering principles.
- Improve our communication skills, with particular emphasis on written communication and, further, well-written programs.
- Practice the appropriate and ethical use of existing material of different kinds, such as source code, services, and documentation.
- Learn how to manage a self-directed project.

Learning Objectives

Students should be able to

- Make effective use of the research literature.
- Determine how available software may be used, subject to both common professional standards and the legal licenses governing the software.
- Choose an appropriate method for contributing their own work (code, documentation, reports) to the profession, including licenses and copyrights that best suit their needs.
- Write code that can be easily used by their peers and others.
- Perform scientifically sound experimental evaluations of their work.
- Evaluate appropriate software engineering methods for individual and team work.
- Present their work in a public forum to their peers and others.

Prerequisites

The three prerequisites for Capstone 2 are Capstone 1, senior standing, and permission of the department chair. *Permission to register will be granted only to those students who have made enough progress in their project work to indicate a high likelihood of timely project completion.* This assessment will be made by the department chair, in consultation with the faculty. A key factor is the recommendation from the project advisor with additional input from the academic advisor.

Students should discuss these prerequisites with their *academic advisors* before seeking help elsewhere. Students with any *special requests* in this regard must address them to the *department chair*, with the support of their academic advisors.

Contact Information

Class meetings:

Time: Tuesdays & Thursdays, 2:00–3:15 p.m.
Location: Neville Hall, Room 204 **or 120**.

Instructor: Sudarshan S. Chawathe

Office: Neville Hall, Room 224.
Office hours: (Please check for changes.) Tuesdays & Thursdays 1:30–2:00 p.m., 3:15–4:00 p.m.
Phone: +1-207-581-3930. Avoid.
Email: `chaw@cs.umaine.edu`. Use email only for messages unsuitable for the newsgroup. (See below.) Please put the string *Capstone* near the beginning of the Subject header of your messages to me.
Web: `http://cs.umaine.edu/~chaw/`.

Online Resources

Class Web site: `http://cs.umaine.edu/~chaw/cap2/`
We will use the class Web site for posting assignments, readings, notes, and other material. Please monitor it.

Class Newsgroup: We will use the local USENET newsgroup `umaine.cs.capstone` on the NNTP server `news.cs.umaine.edu` for electronic discussions. If you are unfamiliar with USENET, you may find the Web interface at `http://cs.umaine.edu/~chaw/news/` useful as a quick way to get started. You may find further information on USENET at `http://en.wikipedia.org/wiki/Usenet`. The newsgroup is the primary forum for electronic announcements and discussions, so please monitor it regularly, and post messages there as well. Unless there is a reason for not sharing your question or comment, please *use the newsgroup, not email*, for questions and comments related to this course.

Class mailing list: *Please make sure you are on the class mailing list.* A sign-up sheet is circulated at the first class meeting. If you miss it, please contact me to get on the list. We will use this mailing list only for urgent messages because all other messages will go on the class newsgroup. I anticipate fewer than a dozen messages on this list over the semester.

Grading Scheme

component	%	
class participation	5	
project reports (versions 1, 2, & 3)	45	(10 + 15 + 20)
source code and demo (versions 1, 2, & 3)	35	(5 + 10 + 20)
final oral presentation	15	

Class participation: Students are expected to contribute to learning by asking questions and making relevant comments in class *and on the class newsgroup*. Quality is more important than quantity. Disruptive activity contributes negatively. Please make sure all disruptive devices are disabled while in class. If you have a good reason for wanting to be disturbed in class, please contact me to make the appropriate arrangements.

Project Reports: The sequence of three project reports serves to systematically document the project. Some details are outlined in the guide for Capstone project proposals (Reading 1). Further details will follow in class. Students are strongly encouraged to continually seek feedback on their working drafts from their project advisors, Capstone instructor, academic advisors, and others.

Source code and demo: Well packaged and documented source code is an important component of the Capstone project. The code will be evaluated on not only how well it functions but also on aspects such as clarity and elegance. The source code does *not* have to be released under any specific license (although a free software license[1] is strongly recommended); however, no legal encumbrances (such as nondiscolsure agreements) will be entertained. All code must be submitted electronically (only) as outlined in the *Submission Instructions* section below.

Final Oral Presentation: Every student must make an oral presentation of his or her work on a date near the end of the semester. The date will be selected to ensure good attendance by department faculty and others, and will be announced in the first few weeks.

Policies

Due dates: All due dates (and times) are strict, as announced in class. If you believe your work was delayed by truly exceptional circumstances, let me know as soon as those circumstances are known to you and I will try to make a fair allowance. However, *the default is that you get a zero if you don't turn in the work on time.*

Attendance: Although I expect students to attend all class meetings, I will not be taking attendance. If you miss a class meeting, you are responsible for making up the lost material. If you have a valid reason for missing a class, let me know early and I will try to help you make up the class.

Make-up classes: I may have to reschedule a few classes due to my other professional commitments. I will make every attempt to minimize the number of such occurrences and to reschedule for a time that works for most students. Further, I will make sure no student is penalized by such occurrences.

Academic honesty (standard university wording): Academic dishonesty includes cheating, plagiarism and all forms of misrepresentation in academic work, and is unacceptable at The University of Maine. As stated in the University of Maine's online undergraduate Student Handbook, plagiarism (the submission of another's work without appropriate attribution) and cheating are violations of The University of Maine Student Conduct Code. An instructor who has probable cause or reason to believe a student has cheated may act upon such evidence, and should report the case to the supervising faculty member or the Department Chair for appropriate action.

[1] such as one compatible with the Debian Free Software Guidelines.

Disabilities (standard university wording): If you have a disability for which you may be requesting an accommodation, please contact Ann Smith, Director of Disabilities Services, 121 East Annex, 581-2319, as early as possible in the term.

H1N1 notice (standard university wording): In the event of disruption of normal classroom activities due to an H1N1 swine flu outbreak, the format for this course may be modified to enable completion of the course. In that event, you will be provided an addendum to this syllabus that will supersede this version.

Readings

This list will be revised and annotated as the semester progresses to reflect, in particular, the topics and papers selected based on class discussions.

1. Sudarshan S. Chawathe. Capstone project proposals—suggestions for deeper explorations. Department of Computer Science, University of Maine. `http://cs.umaine.edu/`, February 2008.

2. Timothy Furtak, José Nelson Amaral, and Robert Niewiadomski. Using SIMD registers and instructions to enable instruction-level parallelism in sorting algorithms. In *Proceedings of the 19th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 348–357, San Diego, California, 2007.

3. Jon L. Bentley and M. Douglas McIlroy. Engineering a sort function. *Software–Practice and Experience*, 23(11):1249–1265, November 1993.

4. Derrick Coetzee. An efficient implementation of Blum, Floyd, Pratt, Rivest, and Tarjan's worst-case linear selection algorithm. `http://moonflare.com/`, January 2004.

5. Bingsheng He, Ke Yang, Rui Fang, Mian Lu, Naga K. Govindaraju, Qiong Luo, and Pedro V. Sander. Relational joins on graphics processors. In *Proceedings of the 28th ACM International Conference on Management of Data (SIGMOD)*, Vancouver, Canada, June 2008. To appear.

6. Naga K. Govindaraju, Jim Gray, Ritesh Kumar, and Dinesh Manocha. GPUTeraSort: High performance graphics coprocessor sorting for large database management. In *Proceedings of the 26th ACM International Conference on Management of Data (SIGMOD)*, Chicago, Illinois, July 2006.

7. Daniel Cederman and Philippas Tsigas. A practical quicksort algorithm for graphics processors. Technical Report 2008-01, Department of Computer Science and Engineering, Chalmers University of Technology and Göteborg University, Göteborg, Sweden, 2008.

8. Sang-Won Lee and Bongki Moon. Design of flash-based DBMS: an in-page logging approach. In *Proceedings of the 27th ACM International Conference on Management of Data (SIGMOD)*, pages 55–66, Beijing, China, June 2007.

9. Gilad Bracha. Generics in the Java programming language. Tutorial. `http://java.sun.com/`, July 2004.

10. Ken Thompson. Reflections on trusting trust. *Communications of the ACM*, 27(8):761–763, August 1984.

11. Mark C. Hamburg. Two tagless variations on the Deutsch-Schorr-Waite algorithm. *Information Processing Letters*, 22:179–183, 1986.

12. Martin E. Hellman. An overview of public-key cryptography. *IEEE Communications Magazine*, 50(5):42–49, May 2002. Originally published in 16(6), November 1978.

13. Jon Bentley and Don Knuth. Programming pearls: Literate programming. *Communications of the ACM*, 29(5):364–369, May 1986.

14. Jon Bentley, Don Knuth, and Doug McIlroy. A literate program. *Communications of the ACM*, 29(6):471–483, June 1986.

15. Paul E. Black. Dictionary of algorithms and data structures. `http://www.nist.gov/dads/`, September 1998.

Assignments, Tests, and Notes

Material will appear here as we move along the semester. It may be useful to refer to the homeworks and tests from the previous session: `http://cs.umaine.edu/~chaw/200901/cap2/`.

Submission Instructions

All electronic submissions must use the file upload interface at `http://cs.umaine.edu/~chaw/u/` with the authentication information announced in class. Uploaded files must be named following the template `cap2-Lastname-Firstname-rep2.jar`, or as announced in class. *No other forms of electronic submission (such as email attachments) are accepted.* Submissions must be properly packaged, with suitable README files, and must contain only *source* code and documentation.

Schedule

An approximate schedule appears in Figure 1. Please use it only as a rough guide to plan your studies. Do *not* use it to schedule travel or other events. If you need a definite answer on when something will or will not occur, you should check with me. The notation Rn refers to the nth item in the reading list.

This schedule will be updated based on the specific topics and readings selected by the class after the first few class meetings.

Tuesday	Thursday
January 12th C1 Introduction; project guidelines; customization; sorting. R1.	14th C2 Customization; sorting and SIMD. R**??**.
19th C3	21st C4
26th C5	28th C6 Project Report 1 due.
February 2nd C7 Source code version 1 due.	4th C8
9th C9	11th C10
16th C11	18th C12
23rd C13	25th C14
March 2nd ×*No class. Spring Break Feb. 27th–Mar. 14th.*	4th ×*No class.*
9th ×*No class.*	11th ×*No class.*
16th C15	18th C16 Project Report 2 due.
23rd C17 Source code version 2 due.	25th C18
30th C19	**April** 1st C20
6th C21	8th C22
13th C23	15th C24 Project Report 3 (final) due.
20th C25 Source code version 3 (final) due.	22nd C26 **Final presentations.**
27th C27	29th C28
May 4th ×*No class. Finals week May 3rd–7th.*	6th ×*No class. Finals week May 3rd–7th.*

Figure 1: ***Approximate*** *schedule, likely to change. All dates, including exam and presentation dates, are tentative!*