FORMAT 1

Submit original with signatures + 1 copy + electronic copy to Faculty Senate (Box 7500).

See http://www.uaf.edu/uafgov/faculty-senate/curriculum/course-degree-procedures-/ for a complete description of the rules governing curriculum & course changes.

TRIAL COURSE OR NEW COURSE PROPOSAL

SUBMITTED BY:

Department	**Computer Science**	**College/School**	**CEM**
Prepared by	**Chris Hartman**	**Phone**	**x5829**
Email Contact	**cmhartman@alaska.edu**	**Faculty Contact**	**same**

1. ACTION DESIRED (CHECK ONE):	Trial Course	x	New Course	

2. COURSE IDENTIFICATION:	Dept	**CS**	Course #	**394**	No. of Credits	3 ~~3~~

Justify upper/lower division status & number of credits:	Will have CS311 as a pre-requisite, requires extensive programming skills and knowledge of higher level CS concepts. Standard lecture course with 3 hours of contact time a week.

3. PROPOSED COURSE TITLE:	**Software Construction**

4. To be CROSS LISTED? *YES/NO*	**NO**	If yes, Dept:		Course #

(Requires approval of both departments and deans involved. Add lines at end of form for additional required signatures.)

5. To be STACKED? *YES/NO*	**NO**	If yes, Dept.		Course #

Stacked course applications are reviewed by the (Undergraduate) Curricular Review Committee and by the Graduate Academic and Advising Committee. Creating two different syllabi—undergraduate and graduate versions—will help emphasize the different qualities of what are supposed to be two different courses. The committees will determine: 1) whether the two versions are sufficiently different (i.e. is there undergraduate and graduate level content being offered); 2) are undergraduates being overtaxed?; 3) are graduate students being undertaxed? In this context, the committees are looking out for the interests of the students taking the course. Typically, if either committee has qualms, they both do. More info online – see URL at top of this page.

6. FREQUENCY OF OFFERING:	**Spring**
	Fall, Spring, Summer (Every, or Even-numbered Years, or Odd-numbered Years) — or As Demand Warrants

7. SEMESTER & YEAR OF FIRST OFFERING (AY2013-14 if approved by 3/1/2013; otherwise AY2014-15)	**Spring 2013**	

8. COURSE FORMAT:

NOTE: Course hours may not be compressed into fewer than three days per credit. Any course compressed into fewer than six weeks must be approved by the college or school's curriculum council. Furthermore, any core course compressed to less than six weeks must be approved by the core review committee.

COURSE FORMAT: (check all that apply)		*1*		*2*		*3*		*4*		*5*	*X*	*6 weeks to full semester*

OTHER FORMAT (specify)	

Mode of delivery (specify lecture, field trips, labs, etc)	**Lecture**

9. CONTACT HOURS PER WEEK:	**3**	LECTURE hours/weeks	0	LAB hours /week	0	PRACTICUM hours /week

Note: # of credits are based on contact hours. 800 minutes of lecture=1 credit. 2400 minutes of lab in a science course=1 credit. 1600 minutes in non-science lab=1 credit. 2400-4800 minutes of practicum=1 credit. 2400-8000 minutes of internship=1 credit. This must match with the syllabus. See http://www.uaf.edu/uafgov/faculty-senate/curriculum/course-degree-procedures-/guidelines-for-computing-/ for more information on number of credits.

OTHER HOURS (specify type)	

10. *COMPLETE CATALOG DESCRIPTION including dept., number, title, credits, credit distribution, cross-listings and/or stacking (50 words or less if possible):*

Example of a complete description:

FISH F487 W, O Fisheries Management
3 Credits Offered Spring
Theory and practice of fisheries management, with an emphasis on strategies utilized for the management of freshwater and marine fisheries. *Prerequisites: COMM F131X or COMM F141X; ENGL F111X; ENGL F211X or ENGL F213X; ENGL F414; FISH F425; or permission of instructor.* Cross-listed with NRM F487. (3+0)

CS F394 Software Construction
Methods for programming and construction of complete computer applications, including refactoring, performance measurement, process documentation, unit testing, version control, integrated development environments, debugging and debuggers, interpreting requirements, and design patterns. Prerequisite: CS 311. (3+0)

11. *COURSE CLASSIFICATIONS:* Undergraduate courses only. Consult with CLA Curriculum Council to apply S or H classification appropriately; otherwise leave fields blank.

H = Humanities		S = Social Sciences	

Will this course be used to fulfill a requirement for the baccalaureate core? **If YES, attach form.**	**YES:**		**NO:**	X

IF YES, check which core requirements it could be used to fulfill:

O = Oral Intensive, **Format 6**		W = Writing Intensive, **Format 7**		Natural Science, **Format 8**	

11.A Is course content related to northern, arctic or circumpolar studies? If yes, a "snowflake" symbol will be added in the printed Catalog, and flagged in Banner.

YES		**NO**	X

12. *COURSE REPEATABILITY:*

Is this course repeatable for credit?	**YES**		**NO**	X

Justification: Indicate why the course can be repeated (for example, the course follows a different theme each time).	

How many times may the course be repeated for credit?		**TIMES**

If the course can be repeated for credit, what is the maximum number of credit hours that may be earned for this course? **CREDITS**

If the course can be repeated with variable credit, what is the maximum number of credit hours that may be earned for this course? **CREDITS**

13. GRADING SYSTEM: Specify only one. Note: Later changing the grading system for a course constitutes a Major Course Change.

LETTER: X **PASS/FAIL:**

RESTRICTIONS ON ENROLLMENT (if any)

14. PREREQUISITES *CS 311*

These will be *required* before the student is allowed to enroll in the course.

Reference the registration implications below due to Banner coding of these terms:
Prerequisite: Course completed and grade of "C" (2.0) or higher prior to registering for the course that requires it.
Concurrent: Course may be taken simultaneously (and allows for a course to have been previously completed).
Co-requisite: Courses MUST be taken simultaneously and does NOT allow for fact that a course was previously completed!

15. SPECIAL RESTRICTIONS, CONDITIONS

16. PROPOSED COURSE FEES **$0**

Has a memo been submitted through your dean to the Provost for fee approval? **Yes/No**

17. PREVIOUS HISTORY

Has the course been offered as special topics or trial course previously? **Yes/No** **No**

If yes, give semester, year, course #, etc.:

18. ESTIMATED IMPACT

WHAT IMPACT, IF ANY, WILL THIS HAVE ON BUDGET, FACILITIES/SPACE, FACULTY, ETC.

No impact further than one faculty member to teach the course and one classroom to teach it in.

19. LIBRARY COLLECTIONS

Have you contacted the library collection development officer (kljensen@alaska.edu, 474-6695) with regard to the adequacy of library/media collections, equipment, and services available for the proposed course? If so, give date of contact and resolution. If not, explain why not.

No	X	Yes		No library resources necessary.

20. IMPACTS ON PROGRAMS/DEPTS

What programs/departments will be affected by this proposed action?
Include information on the Programs/Departments contacted (e.g., email, memo)

None.

21. POSITIVE AND NEGATIVE IMPACTS

*Please specify **positive and negative** impacts on other courses, programs and departments resulting from the proposed action.*

None.

JUSTIFICATION FOR ACTION REQUESTED

The purpose of the department and campus-wide curriculum committees is to scrutinize course change and new course applications to make sure that the quality of UAF education is not lowered as a result of the proposed change. Please address this in your response. This section needs to be self-explanatory. Use as much space as needed to fully justify the proposed course.

See attached.

APPROVALS: Add additional signature lines as needed.

Jon Curtt		Date	8/28/12
Signature, Chair, Program/Department of:	COMPUTER SCIENCE		

Chuen-Lin Lin	Date	09/05/2012	
Signature, Chair, College/School Curriculum Council for:	CEM		
	Date	9/6/12	
Signature, Dean, College/School of:	CEM		

Offerings above the level of approved programs must be approved in advance by the Provost.

	Date	

Signature of Provost (if above level of approved programs)

ALL SIGNATURES MUST BE OBTAINED PRIOR TO SUBMISSION TO THE GOVERNANCE OFFICE

	Date	

Signature, Chair
Faculty Senate Review Committee: ___Curriculum Review ___GAAC

___Core Review ___SADAC

***ADDITIONAL SIGNATURES:* (As needed for cross-listing and/or stacking)**

	Date	
Signature, Chair, Program/Department of:		
	Date	

Signature, Chair, College/School Curriculum Council for:	

	Date	

Signature, Dean, College/School of:	

ATTACH COMPLETE SYLLABUS (as part of this application). The guidelines are online: http://www.uaf.edu/uafgov/faculty-senate/curriculum/course-degree-procedures-/uaf-syllabus-requirements/
The Faculty Senate curriculum committees will review the syllabus to ensure that each of the items listed below are included. If items are missing or unclear, the proposed course (or changes to it) may be denied.

Syllabus CHECKLIST for all UAF courses
During the first week of class, instructors will distribute a course syllabus. Although modifications may be made throughout the semester, this document will contain the following information (as applicable to the discipline):

1. Course information:
θTitle, θ number, θcredits, θprerequisites, θ location, θ meeting time
(make sure that contact hours are in line with credits).

2. Instructor (and if applicable, Teaching Assistant) information:
θ Name, θ office location, θ office hours, θ telephone, θ email address.

3. Course readings/materials:
θ Course textbook title, θ author, θ edition/publisher.
θ Supplementary readings (indicate whether θ required or θ recommended) and
θ any supplies required.

4. Course description:
θ Content of the course and how it fits into the broader curriculum;
θ Expected proficiencies required to undertake the course, if applicable.
θ Inclusion of catalog description is *strongly* recommended, and
θ Description in syllabus must be consistent with catalog course description.

5. θ Course Goals (general), and (see #6)

6. θ Student Learning Outcomes (more specific)

7. Instructional methods:
θ Describe the teaching techniques (eg: lecture, case study, small group discussion, private instruction, studio instruction, values clarification, games, journal writing, use of Blackboard, audio/ video conferencing, etc.).

8. Course calendar:
θ A schedule of class topics and assignments must be included. Be specific so that it is clear that the instructor has thought this through and will not be making it up on the fly (e.g. it is not adequate to say "lab". Instead, give each lab a title that describes its content). You may call the outline Tentative or Work in Progress to allow for modifications during the semester.

9. Course policies:
θ Specify course rules, including your policies on attendance, tardiness, class participation, make-up exams, and plagiarism/academic integrity.

10. Evaluation:

θ Specify how students will be evaluated, θ what factors will be included, θ their relative value, and θ how they will be tabulated into grades (on a curve, absolute scores, etc.) θ Publicize UAF regulations with regard to the grades of "C" and below as applicable to this course. (Not required in the syllabus, but may be a convenient way to publicize this.) Faculty Senate Meeting #171: http://www.uaf.edu/uafgov/faculty-senate/meetings/2010-2011-meetings/#171

11. Support Services:

θ Describe the student support services such as tutoring (local and/or regional) appropriate for the course.

12. Disabilities Services: Note that the phone# and location have been **updated.**

The Office of Disability Services implements the Americans with Disabilities Act (ADA), and ensures that UAF students have equal access to the campus and course materials.

θ State that you will work with the Office of Disabilities Services (208 WHITAKER BLDG, 474-5655)to provide reasonable accommodation to students with disabilities.

8/1/2012

JUSTIFICATION FOR ACTION REQUESTED

In our assessment reports last year, the Computer Science department noted several weaknesses under the following criteria:

C2 - Ability to measure actual performance on a given architecture
C4/K2 - Ability to implement a software system
D2 - Ability to design a large software system (as a group)
D8 - Ability to create software process documents while following a defined process (as a group)
F4 - Ability to create effective software process documents
I1 - Ability to write code without bugs
I3/K3 - Ability to effectively use a version control system to develop software

The current CS catalog does not really have a course that covers these in detail, partly because some of the methods have been developed or greatly extended in the last 10 years and we haven't updated our curriculum. The department agreed (and reported) that for these reasons we should develop a new course as part of our curriculum update. The Assessment report read:

Add CS 372: Software Construction (new course - not offered yet)

- Provide hands-on code performance experience to improve performance for criteria C2. (C2)
- Improve performance for criteria C4. Skills lacking from assessment include: applying design patterns, using version control, designing unit tests, GUI development, Web/back-end development, integration of code from several sources. (C4)
- Cover test planning. (D2)
- Use version control for many assignments. (D8)
- Make changes to existing/open-source code bases and check-in code to improve performance for criteria F4. (F4)
- Write unit tests for code to improve performance for criteria I1. (I1)
- Learn to use a version control system to improve performance for criteria I3. (I3)

We would like to offer this class as a trial course (therefore numbered CS 394) during the Spring 2013 semester, and if successful as a new course (CS 372) starting in Spring 2014.

CS 394 - F01
Software Construction – 3 credits
Spring 2013

Instructor: Dr. Chris Hartman
Email: cmhartman@alaska.edu
Office: 201-D Chapman
Office Phone: 474-5829
Office Hours: TBD or by appointment

Prerequisites: CS 311

Text: *Practical Tools and Techniques for Software Development* by Edward Crookshanks, CreateSpace Independent Publishing Platform; 2nd edition (April 3, 2012)

Course BlackBoard site at http://classes.uaf.edu

Schedule: TBD
Location and Time: TBD

Assessment of the following items will be used in the following proportions to determine student grades.

Assignments	40%
Group Projects	40%
Final Exam	20%

Course description:
From the catalog: CS F394 Software Construction
Methods for programming and construction complete computer applications, including refactoring, performance measurement, process documentation, unit testing, version control, integrated development environments, debugging and debuggers, interpreting requirements, and design patterns. Prerequisite: CS 311. (3+0)

This is a trial course for Spring 2013 which will end up being a required course for all Computer Science students, leading up to the senior capstone sequence of 471/472. In this course we will learn several techniques (see catalog description) for writing large-scale programs that lead to better software with fewer bugs.

The textbook cites the reasons for learning these topics as follows:

The purpose of this companion guide is to discuss and provide additional resources for topics and technologies that current university curriculums may leave out. Some programs or professors may touch on some of these topics as part of a class, but individually they are mostly not worthy of a dedicated class, and collectively they encompass some of the tools and practices that should be used throughout a software developer's career. Use of these tools and topics is not mandatory, but applying them will give the student a better understanding of the practical side of software development.
In addition, several of these tools and topics are the "extra" goodies that employers look for experience working with or having a basic understanding of. In discussions with industry hiring managers and technology recruiters, the author has been told repeatedly that fresh college graduates, while having the theoretical knowledge to be hired, often times are lacking in more practical areas such as version control systems, unit testing skills, debugging techniques, interpreting business requirements, and others. This is not to slight or degrade institutional instruction, only to point out that there are tools and techniques that are

part of enterprise software development that do not fit well within the confines of an educational environment. Knowledge of these can give a student an advantage over those who are unfamiliar with them. This guide will discuss those topics and many more in an attempt to fill in the practical gaps. In some cases the topics are code-heavy, in other cases the discussion is largely a survey of methods or a discussion of theory. Students who have followed this guide should have the means to talk intelligently on these topics and this will hopefully translate to an advantage in the area of job hunting. While it would be impossible to cover all tools and technologies, the ones covered in this guide are a good representative sample of what is used in the industry today. Beyond the theoretical aspects of computer science are the practical aspects of the actual implementation in an enterprise environment; it is this realm that this book attempts to de-mystify. In short, it is hoped that this companion guide will help graduates overcome the "lack of practical experience" issue by becoming more familiar with industry standard practices and common tools. In this volume we cannot create experts, but at least provide enough cursory knowledge such that the reader can discuss the basics of each topic during an interview. With a little practice and exploration on their own, the student should realize that supplementing an excellent theoretical education with practical techniques will hopefully prove useful not only in writing better software while in school, but also translate to an advantage when out of school and searching for a job.

You are expected to be proficient in the material from CS 311 (a pre-requisite) such as advanced C++ programming, common data structures and algorithms, and beginning software engineering techniques.

Expected Student Outcomes:

Ability to measure actual performance on a given architecture
Ability to implement a software system
Ability to design a large software system (as a group)
Ability to create software process documents while following a defined process (as a group)
Ability to create effective software process documents
Ability to write code without bugs
Ability to effectively use a version control system to develop software

Instructional Methods – Classroom lectures, case studies, group presentations.

Group Projects – There will be three group projects consisting of beginning to end development of an application.

Assignments – Assignments will be required generally on a weekly to biweekly basis. The assignments will reinforce lecture concepts and demonstrate application of critical thinking skills. Unless otherwise specified, all assignments must be done on an individual basis. **LATE SUBMISSIONS WILL NOT BE ACCEPTED.**

Policies – Examinations **must** be taken at the scheduled time. In particular, there **will be no** early final exams. You may discuss homework and programming assignments with others, but everything you turn in **must** be your own work.

Disabilities Services – The Office of Disability Services implements the Americans with Disabilities Act (ADA), and insures that UAF students have equal access to the campus and course materials. I will work with the Office of Disabilities Services to provide reasonable accommodation to students with disabilities.

Tentative Schedule:

Week 1: Version control - tools and purpose.

Weeks 2-3: Build tools, automated build engineering, and continuous integration.

Week 4: Debugging - overall summary and introduction to tools.

Weeks 5-6: Unit Testing and Test Driven Development.

Weeks 7-9: Refactoring - purpose and automated tools.

Weeks 10-13: Design patterns and architecture.

Week 13: Documentation and Software Process documents

Week 14: Comparison of development methodologies (waterfall and agile).

Week 15: Interpreting Requirements - business and functional.