

**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	Jon Genetti	Phone	474-5737
Email Contact	<a href="mailto:jdgenetti@alaska.edu">jdgenetti@alaska.edu</a>	Faculty Contact	same

**1. ACTION DESIRED**  
(CHECK ONE):

Trial Course	<input type="checkbox"/>	New Course	<input checked="" type="checkbox"/>
--------------	--------------------------	------------	-------------------------------------

**2. COURSE IDENTIFICATION:**

Dept	CS	Course #	372	No. of Credits	3
------	----	----------	-----	----------------	---

Justify upper/lower division status & number of credits: Will have CS 311 as a prerequisite, requires extensive programming skills and knowledge of higher-level CS concepts. Standard lecture course with 3 hours of contact time per 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:** Every 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 2014

**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.

<b>COURSE FORMAT:</b> (check all that apply)	<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4	<input type="checkbox"/> 5	<input checked="" type="checkbox"/> 6 weeks to full semester
<b>OTHER FORMAT</b> (specify)						
<b>Mode of delivery</b> (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 F372 Software Construction**

**3 Credits Offered Spring**

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. Prerequisites: CS F311. (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:

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

**12. COURSE REPEATABILITY:**

Is this course repeatable for credit? YES  NO

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:  PASS/FAIL:

**RESTRICTIONS ON ENROLLMENT (if any)**

**14. PREREQUISITES**

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**

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

Yes

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

Will be offered Spring 2013 as CS 394

**18. ESTIMATED IMPACT**

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

A faculty member to teach the course once a year and a classroom for the course. The department will use 0.25 FTE saved from suspension of the MSE program.

**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 are 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.

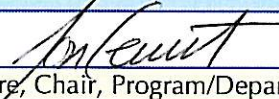
Based on assessment, this will better prepare CS majors for their senior capstone sequence (CS 471/472).

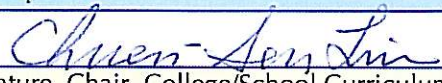
**JUSTIFICATION FOR ACTION REQUESTED**

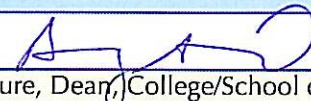
The purpose of the department and campus-wide curriculum committees is to scrutinize course change and new course applications to make sure that the quality of UAF education is not lowered as a result of the proposed change. Please address this in your response. This section needs to be self-explanatory.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

See attached.

**APPROVALS: Add additional signature lines as needed.**

 Date 11/1/12  
Signature, Chair, Program/Department of: Computer Science

 Date 11/06/12  
Signature, Chair, College/School Curriculum Council for: Engineering and Mines

 Date 11/2/12  
Signature, Dean, College/School of: Engineering and Mines

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:		

## ***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)

This has been approved to offer as a trial course during the Spring 2013 semester and we would like to start offering this class as a new course starting in Spring 2014.

**CS 372 - F01**  
**Software Construction – 3 credits**  
**Spring 2014**

**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	30%
Group Projects	40%
Class Participation	20%
Final Exam	10%

**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
- Experience in using design patterns to plan software architecture
- Experience with code reviews

**Instructional Methods** – Classroom lectures, discussion of external readings and case studies and in-class code review, group presentations.

**Class Participation** – You will be expected to participate in discussions of the reading material and case studies, and to actively participate in code-review sessions. Approximately 1/3 of classroom time will be spent on these activities.

**Group Projects** – You'll complete three small software development projects, each of which goes all the way from specification and design to coding and testing. There will be one project with groups of two, one project with groups of three, and one project with larger groups. You'll have the opportunity to choose whom to work with. Any pair of students, however, will only be allowed to work together on a single project. Collaboration is encouraged, although each team member is required to participate roughly equally in every activity (design, implementation, test, documentation, presentation), and I may ask for an accounting of what each team member did. Each project will have multiple due dates, with different deliverables. On the preliminary due dates you will turn documents having to do with exploring the problem and initial design decisions. On the final date, you will turn in final design decisions and working code. Code and design documents will be handed in electronically (by committing in the repository before the deadline). Each project will have at least two in-class presentations (discussing design decisions and finally demonstrating working code.) While each team member will not be required to participate in each presentation, each team member must at least give some part of some presentation.

**Assignments** – Assignments will be required generally on a weekly to biweekly basis. The assignments will reinforce lecture concepts, introduce material needed for group projects, and demonstrate application of critical thinking skills. Unless otherwise specified, all assignments must be done on an individual basis.

**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.



Extremely Tentative Schedule:

Week 1: Version control - tools and purpose.

- Theory
- Resolving conflicts
- Software: Subversion (svn), Git, CVS
- What to keep in a repository

Weeks 2-3: Introduction to the Software Development Process

- Requirements
- Specification
- Architecture
- Design
- Implementation
- Testing
- Debugging
- Deployment
- Maintenance
- Introduction to design patterns
- Modeling languages (UML, etc.)

Week 4: Unit Testing and Test Driven Development

- Theory
- Frameworks
- Automated testing
- Static analysis
- Examples
- Tools for TDD

Week 5: Introduction to Requirements, Project 1

- Stakeholder identification
- Business requirements
- Functional requirements
- Measurable goals
- Prototypes
- Use cases
- Specification
- Project 1 assignment and discussion

Week 6: Build tools, automated build engineering, and continuous integration. The code review process. Project 1 continued.

- Build tools, etc.
  - Make
  - Ant
  - NAnt/MSBuild
  - Maven
  - Continuous Integration tools
  - Examples
- Code review
  - Peer review
  - Automated code review
  - More?
- Project 1 design presentations
- Project 1 work time

Week 7: Debugging and Profiling, Project 1 completion

- Debugging
  - Breakpoints

- Stepping
  - Stack trace
  - Logging
- Performance measurement/profiling/optimizing

Week 7: Refactoring - purpose and automated tools.

- [See Wikipedia's list of refactoring techniques](#)

Week 8: C++11 and Boost

- `shared_ptr<>`
- [support for concurrent programming](#)
- Boost libraries
- [More details here](#)

Week 9: Design patterns, Project 2

- Introduction to design patterns
- [Creational design patterns](#)
- Enterprise patterns
  - Model-View-Controller
  - Inversion of control
- Project 2 assignment and discussion

Week 10: Design patterns, Project 2 continued

- [Structural design patterns](#)
- [Behavioral design patterns](#)
- Project 2 design presentations
- Project 2 work time

Week 11: Software engineering principles, Project 2 completion

- Software Engineering Principles
  - Abstraction principle (programming)
  - Code reuse
  - Single Source of Truth
  - Occam's Razor
  - Separation of concerns
  - KISS principle
  - You Ain't Gonna Need It
  - Rule of three (computer programming)
  - Redundancy (engineering)
  - Mirror (computing)

Week 12: Databases, SQL, web development, Project 3

- Databases
- SQL
- Web front ends
- Web servers
- Project 3 assignment and discussion

Week 13: Anti-patterns, Project 3 continued

- [Anti-patterns](#)
- Code smells
- Project 3 design presentations

Week 14: Documentation and Software Process documents, Project 3 continued

- [Anti-patterns](#)
- Code smells
- Project 3 design presentations

Week 15: Comparison of development methodologies, Project 3 completion

- Development methodologies
  - Waterfall
  - Agile
  - TDD
  - Lean
  - Extreme Programming
  - Iterative
  - Cleanroom
  - Spiral
  - Scrum
  - Incremental
- Project 3 final presentations