CS314: Software Development Methods: Course Introduction

Prof. Robert B. France
Dept. of Computer Science
Colorado State University
My Contact Information

Computer Science Building, Room 458
Office Hours: T&Th 2 to 4 PM or by appointment
Telephone: 970-491-6356
Email: france@cs.colostate.edu

Web: http://www.cs.colostate.edu/~france
CS314 TA Contact Information

Sai Mandalaparty  
Software Engineering Graduate student  
Computer Science Building, Room 252  
Office Hours: TBA  
Email: sai@cs.colostate.edu
Course Highlights

• Course email: cs314@cs.colostate.edu

• Course Syllabus:
  http://www.cs.colostate.edu/~france/CS314/index.html

• Lecture slides and assignments (homework) will be available via RamCT
  – The RamCT blacboard software is being used for the first time this semester; expect some glitches

• Please report any inconsistencies or statements that need clarification on the RamCT site as soon as possible
Examinations

• There will be 3 exams
• All exams are closed book, closed notes
• Examinations must be taken individually
• Make-up examinations will not be given unless it is a medical emergency
• Exam 1: Thurs, Sept. 27, 2012; in class
• Exam 2: Thurs, Oct. 11, 2012; in class
• Final Exam: Dec. 11, 2012, 11:50AM-1:50PM; in class
Programming Assignments

• Objectives
  – Facing reality: Expose students to the realities of “real-world” software development where customer problem descriptions are seldom precise, complete, and consistent
  – Learning how to manage complexity through use of abstractions: “Think before coding”; Develop ability to analyze and synthesize precise descriptions of problems (requirements) and solutions (designs), using appropriate abstractions
Assignment Expectations

• **Do not expect problem descriptions given in assignments to be precise, complete and consistent**
  – The assignments will help you develop skills related to analyzing problem descriptions to identify problems in the statements
  – Creating software models can help students uncover problems with problem statements

• **Assignments are relatively complex – start working on them early!**
  – While the problems are not as complex as some software you will encounter in industry, they are relatively more complex than the assignments you encountered in previous classes. The “code and fix/hack” model of development will not work in this class

• **Getting your program to “work” (i.e., produce correct output) is not enough!**
  – In some assignments other students will attempt to understand and extend your programs. If they have difficulties because you used a bad design then you will lose points
  – You are required to build robust programs: do not assume that only valid inputs to your program. If invalid input is provided your program should respond with an informative error message that also provides feedback on the input expected.
Late Policy

• All homework must be turned in by the deadline stated on RamCT
  – Allows timely grading and discussion of homework
• Exceptions are made in emergency situations
  – Departmental policy for emergency situation applies
  – Discuss situation with me as soon as possible
• There is a standard university exception for university-sponsored events
  – Please make arrangements with me before the event
Prof. R. France Away Schedule

• Sept. 27-Oct. 6: Away in Austria to attend MODELS 2012 conference (to fulfill my duty as Program Chair)

• Oct. 15-17: Away at a research meeting in Washington DC

• During this period design studios or guest lectures are scheduled in class.
Course Resources

• Course syllabus
  www.cs.colostate.edu/~france/CS314

• Using RamCT
Subject Overview

• First course on “programming-in-the-large”
  – Previous courses focused on developing “programming-in-the-small” skills, i.e., on how to put together lines of code that correctly implement a well-understood solution
  – Programming-in-the-large is concerned with developing “large” software systems
    • Problems and their solutions are often not well-understood
    • Requirements often change and thus change must be accommodated

• Focus on developing problem solving skills
  – The complexity of the problems and solutions is such that a single programmer cannot fully understand them
    • Team effort is required
  – “programming-in-the-small” techniques are not enough to tackle the development of complex software systems
Programming-in-the-small skills

• What skills, knowledge did you develop in CS253?
  – Knowledge of programming language syntax?
  – Structured programming skills?
  – OOP concepts?
  – Anything else? (how did you test your programs?)

• Were the problems and solutions you tackled in these courses well-understood?
  – What are some of the problems and solutions you encountered in these courses?

• What happened to the programs you completed?
• Can you distinguish between learning a programming language and learning how to program?
Programming-in-the-large skills

• What do we do when problems and solutions are complex?
  – We tackle the problems and solutions using abstractions.

• Programming-in-the-large is concerned with developing solutions expressed in terms of building blocks that are abstractions of coded implementations.
  – building blocks are sometimes called modules
  – Programming-in-the-small is concerned with putting together solutions using program language statements as building blocks
  – Programming-in-the-large is concerned with putting together solutions using larger building blocks – a single module can be implemented as a single program.
Examples of Large Complex Software Systems

- Space vehicle software
- Mass transportation management systems (air traffic control, train management systems)
- Telecommunication systems
- Financial systems
- Enterprise application systems
- Automobile systems (e.g., drive-by-wire vehicle systems)
- Fly-by-wire aircraft systems
- Social networking software
- Cyberphysical systems (e.g., smart health systems, intelligent transportation systems)
Why is this course relevant?

• Many modern systems are complex.
• Consequences of software errors are more critical today
  – Failures can lead to loss of lives, loss of investments and assets, loss of reputation, loss of privacy
  – Organization survivability often dependent on “mission-critical” software systems
• It is not good enough that systems produce the correct result; they also have to be secure, highly-available, safe, fault-tolerant, evolvable, ...
• Organizations are more interested in software developers that can work in teams to develop software.
Course Objectives

• Develop problem solving skills
  – Develop an understanding of software development principles
  – Develop an understanding of concepts, techniques, and practices that can be used to manage inherent complexity
  – Develop a basic understanding of design qualities and evaluation techniques

• Develop modeling skills
  – Develop an understanding of the role of modeling in software engineering
  – Working knowledge of UML

• Sharpen programming-in-the-small skills
  – Working knowledge of OO design, design patterns, systematic testing techniques
3 Major Topic Areas

• Software Design
  – Modeling-in-the small: Plan your program design before coding
  – Design patterns: Design for change
  – User interface design: Design for usability

• Validation & Verification
  – Focus on systematic techniques for uncovering errors in programs

• Modeling-in-the-large & Software Requirements
  – Focus on how modeling can help uncover problems related to incomplete, ambiguous and inconsistent requirements
  – Modeling problem structure using requirements class models
  – Modeling required behavior using use cases and activity diagrams
The power of models: Supports system thinking
Why the Neanderthals became extinct.
Approach

1. Develop **modeling-in-the-small** skills and enhance programming-in-the-small skills
   - Modeling designs of small programs
   - Systematic testing of programs
   - Design patterns

2. Develop **modeling-in-the-large** skills
   - Modeling large software systems
   - Emphasis on modeling and analyzing requirements
   - Modeling-in-the-large skills are further developed in CS414
Major take away points

• Programming-in-the-small versus programming-in-the-large
  – What is the difference?
  – Why do we need to develop these skills?
• Modeling-in-the-large versus modeling-in-the-small
  – What is the difference?
  – Why build abstract models of software?
Staying Ahead

• Start assignments at most 1 day after it is made available.
  – Assignment is NOT designed to be started and completed overnight!

• Keep ahead of the reading material. Stay ahead by reading the relevant sections of the text book before the class.
  – Weekly reading assignments are available on RamCT.

• Students are strongly encouraged to read more than just the course notes. You cannot get a B or above in this class if you read only the lecture slides.
Reminder: Choose your assignment partner this week!

After you have selected a partner send email to cs314@cs.colostate.edu with your name and email address and your partner’s name and email address.
“Learning a programming language is (relatively) easy, learning how to program is difficult” - unknown

“The sooner you start to code the longer it will take to complete the program” – Ray Carlson