CS 4374/5374 Software Construction

Course Syllabus

Fall 2018

General Information

- *Course title*: CS 4374/5374 – Software Construction
- Semester and sequence number: Fall 2018, 1826 & 16942.
- Time and Location: M 6:00 PM-8:50PM. CCSB G.0208
- *Instructor*: Dr. Omar Badreddin. Email: obbadreddin@utep.edu
- *Office location*: 3.0610.
- *Office hours*: MW 3:00 PM – 5:00 PM.
  By appointment as needed. I am in my office all day with the door open, you are encouraged to drop in any time.

Course Restrictions

- Restricted to majors of CS and SWE.

Course Description

Survey of professional software construction techniques and practices including agile development, software tools and environments, configuration management, defect tracking, coding style, coding standards, cross-compilation, techniques for optimization (time, space, and I/O bandwidth), refactoring, software maintenance, and software development automation. Provides an integrated view of subjects related to the different phases of software development.

Student Learning Expectations/Outcomes for this Course

The goal of this course is to teach students advanced software design techniques, and expose them to some of the cutting-edge design and modeling tools. Students should gain an understanding of the modeling practices of software engineers that ranges from model centric to code centric approaches. The students should have an understanding of model driven engineering concepts. Students will have some applied experience in developing sufficiently large systems using model driven methodology. The students will experience researching an advanced topic and write a short report paper.

In specific, the primary objectives of the course are to:

- Understand the process and techniques of designing large software systems
- Appreciate the challenges involved in the design and development of large software systems
- Gain deep understanding and practice with class diagrams, state machines, and sysML modeling notations.
- Understand and practice the techniques of forward and reverse engineering
- Understand and practice automated program generation
- Understand some of the current research topics in the area of model driven engineering

- Communicate (written and verbally) about a complex, technical topic simply and coherently.
- Be able to work and interact collaboratively in groups to examine, understand and explain key aspects of advanced software design.
- Be able to do research, write and present on a current topic in the domain of modeling
- Be able to work with guiding and managing undergraduate teams doing research on software design topic
- Be able to present and communicate at the graduate-level including the use of an approved writing style for research topics

Assessment of Student Learning Outcomes

In addition to the traditional assessment tools such as exams, assignments, and quizzes, this course includes a research topic presentation. This involves students selecting a topic of interest, collect and analyze the related literature. A list of topics will be presented

The grading breakdown is as follows:

Assignments (~3 to 5 assignments)	20%
Attendance, Participation, Quizzes (Announced)	10%
Midterm	15%
Research/Project Presentation	15%
Research topic/paper Report	15%
Second Midterm	25%

Grading scale: A = [90—100], B= [80—90), C=[70—80), D=[60—70), and F=[0—60].

Course Structure and Approach

The material discussed in the lecture will be both theoretical and practical. New concepts and theories will be backed up by live demos and illustrations. The students will have a chance to apply the new concepts to solve problems in the assignments.

Textbooks and required Materials

There is no required textbook for this course. The students will be provided with handouts and referred to material available online. Such material will be posted on the course BBlearn.

A good reference for many topics discussed in this course is: Software Engineering Theory and Practice" Fourth Edition. Shari Pfleeger and Joanne Atlee.

Recommended optional Materials/References

Referenced Resources

- UML Distilled: A Brief Guide to the Standard Object Modeling Language, Martin Fowler. Addison-Wesley Professional; 3 edition. ISBN-10: 0321193687
- Object-Oriented Software Engineering: Practical Software Development using UML and Java, Second Edition. Timothy C. Lethbridge and Robert Laganière. McGraw Hill, 2001. ISBN 0-07-710908-2

- UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design (2nd Edition), Jim Arlow. ISBN- 0321321278.
- The Umple user manual: http://cruise.eecs.uottawa.ca/umple/

Additional Readings (Research articles)

- Badreddin, Omar, Timothy C. Lethbridge, and Maged Elassar. "Modeling Practices in Open Source Software." Open Source Software: Quality Verification. Springer Berlin Heidelberg, 2013. 127-139.
- Omar Badreddin, Timothy C. Lethbridge, "Model Oriented Programming: Bridging the Code-Model Divide". ICSE Workshop on Modeling in Software Engineering, 2013.
- Omar Badreddin, Andrew Forward, Timothy C. Lethbridge. Model Oriented Programming: An Empirical Study of Comprehension. In proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research.
- Weilkiens, Tim. Systems engineering with SysML/UML: modeling, analysis, design. Morgan Kaufmann, 2011.
- Nugroho, Ariadi, and Michel RV Chaudron. "The impact of UML modeling on defect density and defect resolution time in a proprietary system." Empirical Software Engineering (2013): 1-29.
- Nugroho, Ariadi, and Michel RV Chaudron. "Evaluating the impact of UML modeling on software quality: An industrial case study." Model driven engineering languages and systems. Springer Berlin Heidelberg, 2009. 181-195.

- Omar Badreddin, Andrew Forward, Timothy C. Lethbridge. Model Oriented Programming: An Empirical Study of Comprehension. In proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research.
- Weilkiens, Tim. Systems engineering with SysML/UML: modeling, analysis, design. Morgan Kaufmann, 2011.
- Nugroho, Ariadi, and Michel RV Chaudron. "Evaluating the impact of UML modeling on software quality: An industrial case study." Model driven engineering languages and systems. Springer Berlin Heidelberg, 2009. 181-195.

Suggested Software

In this course, students will need to use advanced modeling tools for class diagrams, state machines, and SysML. The students are free to apply their tool of choice. The following are recommended tools.

- For Class and state machine models: Umple Modeling tool, available online at http://try.umple.org
- For SysML, Visual studio is sufficient, since we will not be generating code from SysML models.

Course Outline

Topic	Name	Description	Duration (weeks)
A	Overview of Software Development Processes	Presentation of key software development processes with focus on agile methodologies.	**1**
B	UML Review & UML Extension Mechanisms	Introduction of the different modeling notations of UML. An overview of UML profiles and UML extension mechanisms.	**1**
C	Advanced UML Modelling	Class and state machine modeling. Presentation of associations, multiplicities, simple and composite state machine models, and related code generation fundamentals, techniques, and tools.	**1**
D	Meta-modelling	Introduction of the concept of meta-modeling, and meta-meta modeling.	**0.5**
E	OCL	Introduction of the OCL constraints language.	**0.5**
F	SysML	System modeling using SysML.	**0.5**
G	Program Refactoring and Comprehension	Introduction to refactoring and program comprehension techniques, software maintenance and analysis.	**2**
H	Code Smells	Discussion of various coding styles, code smells, practices and tools for evaluating code quality and complexity.	**2**
I	Technical Debt	Introduction and discussion of Technical Debt concepts and implications, and quantification methods.	**0.5**
J	Software Configuration Management	Discussion of the fundamental practice of Software Configuration Management, best practices, tools, and techniques, defect and feature tracking.	**2**
K	Blockchain Fundamentals and Development Platforms	Introduction to Blockchain fundamentals. Overview of some key open source blockchain development platforms.	**1**

List of Research/Paper Topics

Research topics are listed below. Topics are broad and multiple groups may choose the same topic. However, I will encourage unique topics to maximize learning. Additional information on each topic and requirements of the presentation and report to be discussed in class.

I. **Model Based Testing**
   This topic covers methodologies, tools, and techniques where software testing is performed on software models (or possibly on code generated from models or mix of both).

II. **Forward Engineering**
   This topic covers methodologies, tools, and techniques related to the automated (or semi-automated) code generation (i.e. from models to code). This include visual and textual modeling notations.

III. **Reverse Engineering**

This topic covers methodologies, tools, and techniques related to the automated recovery of software designs and models from existing software code. This include reverse engineering of executables to recover originating source code or pseudo code.

IV. Constraints Modeling and OCL

Object Constraints Language (OCL) has been discussed in class. This is a related topic that addresses other constraints modeling notations in addition to OCL. This topic also covers methodologies and tools related to constraints modeling in software and software-intensive systems (i.e. including cyber-physical systems).

V. Challenges related to Code-Centric Software Construction

The majority of software systems today are developed following a code-centric approach. This topic covers issues related to challenges in code-centric development (i.e. issues related to the cost of software development, quality and reliability of software systems, software systems evolution and maintenance, etc.)

VI. Program and code comprehension

This topic covers studies that evaluates how software engineer understand code, aspects of code that improves comprehensibility, as well as issues related to software maintenance and evolution. Particularly, code analysis tools (both static and dynamic), tools and techniques to assess code quality and sustainability, tools to identify and quantify code smells and Technical Debt.

VII. Programming Language Design

This topic covers aspects related to the tools and methodologies of language design, including domain specific language (DSL) design. This also includes both visual and textual languages.

VIII. Modeling Cyber-Physical Systems

This topic covers aspects related to the design and modeling of software and software-intensive systems (systems where software and hardware share strong interdependencies, i.e. cyber-physical systems).

IX. Blockchains and Development platforms

Some suggested Blockchain development platforms include IBM Blockchain, OpenChain, Quorum, and others.

Assignment Description (tentative)

The following is a description of each assignment

<u>Assignment 1 (UML)</u>

This includes exercises on class diagrams, state machine modeling, and SysML. The questions ask the student to model a part of the system using each of the modeling notations. The students are exposed to modeling complex data (class diagrams) and behavior (state machine) and the entire systems (SysML).

The objective of this assignment is two folds. First, to ensure that students have a hands on practice with the concepts discussed in the lecture. Second, to help the students appreciate the value of modeling.

Assignment 2 (Meta-modeling, OCL, and Aspect Orientation)

This assignment gives the students an exercise to create a meta-model for a specific domain. They are asked to restrict a class diagram by writing a number of OCL statements. Questions on Aspect Orientations are theoretical.

The objective of this assignment is 3 folds. First, practice the concepts discussed in class. Second, understand the power of meta-modeling and OCL statements. Third, be able to articulate the distinction between associations, meta-models, and OCL constraints.

Assignment 3 (forward and reverse engineering)

This assignment focuses on code generation for all the modeling notations discussed in class. The assignment asks the students to provide a hand written implementation for some modeling samples and compare that to code that is automatically generated by a compiler. Graduate students are asked to provide a critic for the differences. Similarly, the students are asked to reverse-engineer an existing system and provide the model. Graduate students are asked to compare this to an automated approach using a tool of their choice.

The objective of this assignment is to give the students hands on experiences with forward and reverse engineering, and understand the limitations with current automated software generation techniques.

Course Policies

Exams and Quizzes

Without prior notice of illness or documented substantial extenuating circumstances, there are no make-ups for exams and quizzes. Please be prepared to provide supporting documentation to substantiate circumstances, as needed.

Attendance

It is critical that you attend every lecture and group meeting. In this course, the readings and lectures will go hand-in-hand and will not necessarily cover the same material. One will reinforce the other, and -- to do well -- you should be prepared to come to lecture having read through the reading assignments for the day. Exam content will be drawn from both lectures and readings.

Please be cautious about attending class and meetings if you are feeling ill. Please inform me by email if you are feeling unwell; if you are experiencing flu-like symptoms, you should not attend class and seek medical attention.

Excessive absence can result in an automatic F in the course.

Class Conduct

Appropriate in-class student conduct is a critical component of a smoothly running course. Please be courteous in your interactions with me and other students and ensure that your behavior supports a

positive learning environment and is not disruptive to the normal flow of the course. Examples of disruptive behavior include, but are not limited to, the following:

- Showing up late to class;
- Preparing to leave before the instructor has dismissed the class;
- Maintaining conversations with neighboring classmates at inappropriate times;
- Speaking without being recognized, asking questions or making comments irrelevant to course material; Interrupting the instructor or other students;
- Being obviously disengaged or disinterested in the subject matter;
- Refusing to comply with an instructor's request;
- Making calls or holding text-message conversations using your cellphone;

These rules of conduct apply to any online discussions (such as BBLearn) used in the course.

All that said: Healthy discussion, at times permitted by the instructor, is highly encouraged!

Late Submissions

Assignments are to be submitted on BBlearn. Late assignments are not accepted.

Electronic Devices

Feel free to bring your laptops and take electronic notes or try things out as we talk about them during lecture. Note that watching YouTube videos or updating your Facebook page does not count as taking notes and trying things out. Please be courteous to your classmates and me by silencing your cell phones. I reserve the right to ask you to stop using any device if I feel its use is bothersome or distracting to the class.

Contact Methods

Please don't hesitate to drop by my office or send me an email with any personal concerns. I will happily do my best to answer your questions and address your concerns. I reserve the right to ask you to come in for a chat during office hours for long answers, and reserve email for shorter answers. I will answer your emails as soon as I possibly can, but don't bank on a response time measured in minutes (though, that may sometimes happen too). Please make sure that you put your name and course number somewhere in the message.

Academic Integrity

One of the foundations of academic life is honesty. Assignments and exams are ways to measure your understanding of the material being covered in the course, not medieval implements of torture. By cheating, you are cheating yourself out of the chance to have your understanding accurately evaluated. Grades are an indication of your final proficiency over the material, and not a form of punishment. Be honest and fair to your fellow classmates: do your own work. You'd also be surprised at how easy it is to

spot cheating.

Cheating and any other form of academic dishonesty will be dealt with seriously. Consequences to incidents of academic dishonesty may include a zero grade in the assignment in question, an F in the course, or may be referred to the university's channels and result in expulsion from UTEP -- any and all at my discretion.

Just don't do it!

University Policies

This course is conducted in accordance with all applicable university policies.

Disability and Special Accommodations

If you have or suspect a disability and need accommodation you should contact CASS at 747-5148 or dss@utep.edu or visit Room 106 Union East Building.