CSE 470E/570E: Intro to Model-Driven Software Engineering

Course Description: An introduction to model-driven engineering (MDE) techniques; applying software engineering practices to model-based artifacts; modeling and abstraction; model-based testing; model-checking; tool implementations.

Prerequisite: CSE 201 (Intro to Software Engineering)

Lecture Slot: Mondays and Wednesdays, 11:40am – 1:00pm, Benton Hall 009

Who Takes This Course: This course is an elective for CS Majors, and can be used in SE Specialization Areas.

Contact Information

Instructor: Dr. Eric J. Rapos (rapose@miamioh.edu)

Office Hours in Benton 205D:

- Tuesdays TBD
- Wednesdays TBD
- Thursdays TBD

Student Rights & Responsibilities

As a student in this course, you have the right to:

- provide feedback to the instructor about course delivery and timing of deliverables;
- provide feedback to the instructor about instructional techniques and other comments on instructor ability, both via structured evaluations and independently;
- attend office hours (and other times as available) to discuss course content with the instructor;
- receive grades for assigned work within one week of submission;
- request additional feedback and justification for any grade received through the course; and
- request extensions to deadlines in extenuating circumstances – note that extensions will not always be granted.

As a student in this course, you are responsible for:

- actively participating in course discussions;
- being respectful to your classmates and of their opinions;
- informing the instructor of any conflicts with test and other deliverables at least two weeks in advance;
- reporting to the instructor any issues of an academic integrity nature;
- completing all assigned work by the informed deadline; and
- asking for assistance from the instructor when needed.

Student Learning Outcomes

1. Explain key differences between standard software engineering and model-driven engineering practices.
    1. Explain the process differences beginning with the emphasis on models as primary artifacts.
    2. Be able to identify similarities between known SE practices and counterparts in MDE, and discuss how they differ.
2. Explain and be able to create meta-models and resulting instance models for typical software systems.
    1. Explain how meta-models differ from instance models.
    2. Construct a meta-model for standard examples and subsequently the instance models that correspond.
3. Explain the code generation process.
    1. Be able to identify the corresponding aspects in an instance model and the resulting generated code.
    2. Demonstrate the uses of simulation as a precursor to code generation.
4. Explain dynamic/real-time modeling.
    1. Understand the differences between standard models and real-time behavioral models.
    2. Create UML-RT representations of simple real life systems.
5. Explain validation using models as primary artifacts.
    1. Demonstrate effective model-based testing for standard and model-based software systems.
    2. Understand using model-based testing as a means of validating both simulation and generated code.
6. Demonstrate proficiency in creating simple models in current modeling technologies.
    1. Create simple UML-RT models.
    2. Create simple Simulink behavioral models.
    3. Create simple EMF models.

Copyright Disclaimer

Disability Services

If you are a student with a physical, learning, medical and/or psychiatric disability and feel that you may need a reasonable accommodation to fulfill the essential functions of the course that are listed in this syllabus, you are encouraged to contact the Miller Center for Student Disability Services at 529-1541 (V/TTY), located in the Shriver Center, Room 304.

Mental Health Services

If you are a student who may be experiencing mental or emotional distress, you are encouraged to call Student Counseling Service (513-529-4634). For emergencies outside of business hours, the Community and Counseling and Crisis Center (844-427-4747) has a 24-hour hotline.

Academic Support

The following resources are available for you as a student:

- Rinella Learning Center Academic Support
- Howe Center for Writing Excellence
- International Student Resources
- Student Success Center

Class Attendance

All students are required to attend every class session, however, no grades will be assigned for attendance. Wednesday Class sessions have an associated deliverable, and as such your attendance will be noted through the completion of these items (labs and tests).

Refer to the Miami University Policy on Student Attendance for full details on the university policy on attendance.

Grade Conversions

From %	To %	Letter Grade
96	100	A+
93	95.99	A
90	92.99	A-
86	89.99	B+
83	85.99	B
80	82.99	B-
76	79.99	C+
73	75.99	C
70	72.99	C-
66	69.99	D+
63	65.99	D
60	62.99	D-
0	59.99	F

Important Dates

- Last Day to Drop the Course with No Grade
  - Thursday Feb 14, 2019
- Last Day to Withdraw with a Grade of W
  - Monday, April 8, 2019

Required Course Textbook

Model-Driven Software Engineering in Practice: Second Edition (Synthesis Lectures on Software Engineering)
Paperback: 208 pages
Publisher: Morgan & Claypool Publishers; 2nd edition (March 30, 2017)
Language: English
ISBN-10: 1627057080
ISBN-13: 978-1627057080

Course Grading

Your grade in this course will be based on the following deliverables:

Deliverable	Percentage of Final Grade 470E	570E	Due Date
Individual Assignments			
Assignment 1: Eclipse Modeling Framework	5	4	Feb 22 (F)
Assignment 2: Modeling as a Primary Artifact	5	4	Mar 8 (F)
Assignment 3: Papyrus-RT & Simulink Modeling	5	4	Apr 12 (F)
Assignment 4: Model-Based Testing	5	4	Apr 26 (F)
Labs			
Lab 1: Meta-Models	2	2	Feb 6 (W)
Lab 2: Eclipse Modeling Framework	2	2	Feb 13 (W)
Lab 3: Models as a Primary Artifact	2	2	Feb 27 (W)
Lab 4: Model Transformations	2	2	Mar 6 (W)
Lab 5: Behavioral Modeling	2	2	Mar 13 (W)
Lab 6: Papyrus-RT	2	2	Apr 3 (W)
Lab 7: Simulink	2	2	Apr 10 (W)
Lab 8: Model Based Testing	2	2	Apr 17 (W)
Lab 9: Papyrus-RT Visualization	2	2	May 1 (W)
Tests and Exams			
Test 1	5	5	Feb 20 (W)
Test 2	5	5	Mar 20 (W)
Test 3	5	5	Apr 24 (W)
Final Exam	20	20	May 17 (F)
Course Project Phases			
Phase 1: System Selection	2	1	Mar 1 (F)
Phase 2: System Meta-Model	4	3	Mar 15 (F)
Phase 3: System Instance Model	4	3	Apr 5 (F)
Phase 4: System Behavior Implementation	4	3	Apr 19 (F)
Phase 5: Model Based Testing	4	3	May 3 (F)
Phase 6: Class Presentation	4	3	May 8 (W)
Canvas Reading Quizzes			
Quiz 1: Chapters 1, 2, 3	1	1	Feb 1 (F)
Quiz 2: Chapters 5, 6	1	1	Feb 8 (F)
Quiz 3: Chapters 8, 9	1	1	Feb 15 (F)
Quiz 4: Chapters 4, 7	1	1	Mar 22 (F)
Quiz 5: Chapter 10	1	1	May 10 (F)
Graduate Research Work			
Graduate Research Paper	--	10	May 10 (F)
TOTAL	100	100	

Each item will be graded out of a set number of points (rubrics and details will be provided as each item is assigned), and then scaled based on the weights provided above. Your final grade will not be shown in Canvas, but you may request your current performance from the instructor at any time, or calculate it using the above weights.

Academic Integrity

The Department of Computer Science and Software Engineering is committed to maintaining strict standards of academic integrity. The department expects each student to understand and comply with the University's Policy on Academic Integrity and the undergraduate student handbook and graduate student handbook. Students may direct questions regarding academic integrity expectations to their instructor or to the department chair. All work submitted must be original for that class. Submitting the same project for two different classes is grounds for charging a student with academic misconduct unless prior written permission is received from both instructors.

"Problem Solving Assignments" are assignments that involve programming, math, proofs, derivations, and puzzles. The purpose of a problem solving assignment is for you to develop the skills necessary to solve similar problems in the future. To learn to solve problems you must solve the problems and write your solutions independently.

It is worth reiterating that the important aspect of the assignment is that you actually create the solution from start to finish; simply copying a solution and then understanding it after the fact is not a substitute for actually developing the solution.

The notion of academic integrity can be confusing in courses with substantial problem solving because certain forms of collaboration and investigation are permitted, but you are still required to complete your assignment independently. The following scenarios are meant to help distinguish between acceptable and unacceptable levels of collaboration and research, but are not all-inclusive:

ACCEPTABLE:

- Consulting solutions from the current course textbook, but not from other published sources.
- Seeking help on how to use the programming environment such as the editor, the compiler, or other tools.
- Seeking help on how to fix a program syntax error or how a certain language feature works.
- Discussing strategies with a fellow student on how to approach a particular problem. This discussion should not include significant sections of completed work or source code (including printouts, email, viewing on a monitor). Discussions should begin with a clean sheet of paper and end with conceptual drawings and/or pseudo-code.

UNACCEPTABLE:

- Looking at another solution including those written by current students, past students, or outside sources such as code or solutions found on the Web, or in publications other than the current class textbook.
- Using another solution as a starting point and then modifying the code or text as your own work.
- Providing a copy of your solution or a portion of your solution, in any form (electronic, hard copy, allowing another student to view your code on a monitor), to another student.
- Giving or receiving code fragments to fix a problem in a program.
- If you are stuck on a problem and you are tempted to search for a solution on the Web or to look at another student's solution STOP and email or ask your instructor for help.