UTC-IASE
UNIVERSITY OF CONNECTICUT

UCONN
SCHOOL OF ENGINEERING

SE 5201: Embedded/Networked Systems Modeling Abstractions

Course Instructor: **Fei Miao, Ph.D.**

Catalog Description. 3 credits. Students will become cognizant of the role of embedded controllers and devices in the system design process, as they relate to event-driven and data-driven systems, and supervisory control of hybrid (continuous and discrete-time) systems. This will include exposure to platform-based design principles with an emphasis on requirements capture and refinement to platform architecture mapping, analysis and verification. Students will learn the technical aspects of modeling principles relevant to embedded systems – specifically modeling system architecture, system functions, computation, software, real-time systems, and distributed systems. Use of software engineering tools (Rhapsody, Simulink, Stateflow and Simulink/MATLAB coder) in the embedded system design flows is emphasized.

Pre-Requisites Background in hardware and/or software design.

Intended Audience. The course is addressed to students who have had at least a year of software or embedded systems design experience and who are pursuing a graduate engineering degree or certificate.

Course Delivery Method. The course will be offered online, asynchronously, in small recorded modules according to the course schedule and syllabus. Direct and live communication with the instructor will be available each week, according to the class schedule, for discussion, questions, examples, and quizzes. Attendance at live sessions is required, and you must notify the instructor in advance if you cannot attend. A social networking tool called Slack will be used to communicate with students and the instructor between live sessions.

Course Objectives. The goals of this course are to familiarize students with designing, implementing and verifying embedded systems, and to provide skills necessary to specify requirements and perform platform-based design, analysis and modeling of embedded and networked systems. These models will be motivated by applications from industry which demonstrate embedded systems design challenges of satisfying time-critical, event-driven, and data-centric requirements.

Anticipated Student Outcomes. By the end of the course, a student will be able to:

(1) Learn what embedded systems are, what is desired and what can typically go wrong in embedded system design and implementation.

(2) Understand how to formulate and model embedded system requirements.
(3) Learn how to analyze and map requirements into embedded system architectures.
(4) Learn how to model system architectures, including heterogeneous systems, using a system modeling language, such as SysML for architecture analysis and design.
(5) Understand fundamental principles of finite state machines and their use in modeling embedded systems for time-critical, event-driven and data-centric systems.
(6) Learn the principles of modeling computation and functional units.
(7) Learn the principles of object and software modeling (using UML) and automatic code generation.
(8) Learn the basic concepts of real-time operating systems and real-time task models.
(9) Learn basic concepts of distributed systems modeling.

Course Organization. The course is organized into six learning modules:
(1) Product development processes and robust design
(2) Concept sizing and margin analysis
(3) Uncertainty quantification and sensitivity analysis in design
(4) Capability analysis
(5) Robust design
(6) Root cause analysis with models

Course Outline. The structuring of these (3) three learning modules into 14 lectures of a one semester course, along with the topics and references, is described in the following.

--------Module I: Design for Embedded System Design, Implementation & Verification--------

This module is meant to offer a general introduction to the course, objectives and key concepts that will be studied. It is comprised of three lectures devoted to a general introduction, to understanding the challenges that arise when modeling embedded systems and finally to the existing product development processes for embedded systems. The third lecture is meant to elaborate on 3 use-cases that will be used throughout the remainder of the course and that exemplify the types of challenges faced in practice. It is also meant to provide a roadmap to navigate the subsequent lectures and remind the students of the mapping between lectures and use-cases, challenges being addressed in that use case, and the methodological underpinnings. The intent is to rely on use cases from the relevant industry sectors. The challenges being addressed are focused on three classes of embedded systems that are (1) time-critical (2) event-driven and (3) data-centric.

Lecture 1: Systems Engineering Fundamentals

- Systems Engineering: DoD, IEEE STD 1220, INCOSE, NASA, US DOT.
- Management and Technical Tasks, Integrated Product Database supports Viewing, Authoring, Synthesis, Analysis.
- DoD Architectural Framework of Views (DoDAF), VEE Method: Synthesis, Analysis, Verification.
- Methodology Control: Waterfall, Iterative, Prototyping, Exploratory, Spiral, Reuse, Agile/Scrum, XP.
- Reuse: As-Is, Minor Changes, Major Changes, Extension, Severe Changes.
- Quality Assurance: CMMI, TQM, Six Sigma, Lean, ISO-9000.
- Development of Customer Requirements, Product Requirements, Embedded System Requirements.

Lecture 2: Systems Engineering in Industry – Part 1

- Enterprise Business Modeling: Marketing and the Virtual Customer, Contracting and the Specific Customer.
- Products of interest in industry.
- Project Management in industry: PASSPORT, SIMILAR, Phase Gate.
- Methodology Control in industry: Platform Based Design, VEE Design Flow.

-------Module II: Embedded Systems Requirements Capture and Architecture Selection--------

This module is devoted to understanding platform-based design as a comprehensive methodology for embedded system design. This module is comprised of three lectures focused on the presentation of PBD as a process for matching the top-down approach starting from the application requirements and the bottom-up approach from the platform selection. The purpose of a platform-based design is not only to leverage libraries of reusable and generic models and components, but more importantly enable designers to answer questions on the feasibility, performance and suitability/viability of the selected platform with respect to key requirements. The second lecture in this module addresses the gathering of requirements for the embedded system design process. It focuses on the specific aspects of the system that must be part of the information gathering process (e.g., performance, reliability, real-time guarantees, etc.). The third lecture of this module focuses on the bottom-up part of the process and the architecture selection, in particular. It discusses how to characterize the platform specifics and how to approach the problem of deciding which functions should be implemented in hardware and which ones should be supported in software.

Lecture 3: Embedded System Requirements

- Requirements Development: Customer Requirements, Product Requirements, Embedded System Requirements
- VEE Process: Platform Based Design, Reuse, Synthesis, Analysis, Trade Study, Verification
- Design Iteration Phase, Design History Tree, Selection of Components Relevant to Embedded Systems

Lecture 4: Embedded System Design Considerations

- Processor: Speed, Memory, Instruction Set
- Real Time: Hard, Firm, Soft, None
- Networking: Congestion, Latency, Throughput, Protocol
- Sensors, Actuators: Analog, Digital, Serial, Parallel
- Architecture: RTOS, Middleware, GUI
- RTOS: Scheduler Type, Scheduling Strategy
- Algorithms: Control, Processing, Event Handling, GUI

Lecture 5: Platform-based design

Learning Objectives: Students will learn about the basic concepts of platform-based design, including embedded systems characteristics, motivations, and keys to effective platform-based design.

- Systems Engineering and where do embedded systems fit in?
- Characteristics of embedded systems that make design hard
    - Constrained resources
    - Concurrency
    - Non-determinism
    - Real-time constraints
- Introduction to platform-based design
    - Motivation
    - Meet-in-the-middle-approach
        - Top-down and Bottom-up
        - Mapping of required behavior to components
- Keys to platform-based-design
    - Models
    - Components
    - Composition, refinement, and abstraction rules
    - Contracts

----------Module III: Modeling Methods----------

This module dives into the technical aspects of the modeling process and initiates the analytical aspects of the course. It is comprised of 7 lectures focused on modeling the entire system, functional unit modeling methods and tools, software modeling and code generation, real-time architectures and operating systems, and finally distributed system issues. Structuring of these 3 learning modules into 14 lectures of a one semester course, along with the topics and references, is included below. In the following, each entry labeled "Lecture" corresponds to 2h: 30m of teaching (either 2 periods of 1h: 15m or a single period of 2h: 30m).

Lecture 6: Finite State Machine Models

Learning Objective: Provide students with fundamental knowledge of finite state machine representations needed for modeling, analyzing and design of embedded systems. At the end of this lecture, students should be able to (1) understand and be conversant with basic mathematical foundations of finite state machines, regular expressions, different types of FSM and their equivalences, (2) Design optimal FSM with different levels of complexity to model embedded systems, (3) Able to Use state charts to represent hierarchical FSM with extended capabilities, such as concurrency.

- Definitions
  - Model of Computation
  - Abstract machine
  - Finite State Machine (FSM)
- FSM Classifications
  - Deterministic (DFA) vs non-deterministic (NFA)
  - Synchronous vs asynchronous
  - Moore vs Mealy
- FSM Views
  - Black box view
  - Network view
- FSM Optimization
  - State Minimization
  - State Equivalence
- Equivalence between NFA and DFA
  - Conversion of NFA to DFA
  - Examples
- Using FSM as a Model
  - Hierarchical FSM

- Concurrency
- Mealy-Moore Solution
- State Chart Solution
- Harel State Chart
- History Mechanism, Default Mechanism and their combination

Lecture 7: Function Modeling/Models of Computation

Learning Objective: Understand the basic principles of system and architecture models. The intent is that students understand the basics and associate them with the practice through some simplified examples. Eventually, they are expected to apply these in practice.

- System Model
  - What is a system model?
  - Why do you need system models?
  - Topics on system modeling
  - Modeling continuous behaviors
  - Modeling discrete behaviors
  - Hybrid systems: continuous & discrete behaviors
  - Composition of state machines
  - Concurrent models of computation
    - Semantics of concurrent composition
    - Components in concurrent composition
    - Structure of models
    - Synchronous-Reactive models
    - Dataflow models of computation
    - Timed models of computation
- Architecture Model
  - What is an architecture model?
  - Basic Architectural Elements
  - Why model architectures?
  - Important characteristics of architecture models
  - Evaluating Modeling Approaches
- Modeling languages o Generic approaches
  - Early architecture description languages
  - Domain- and style-specific languages
  - Extensible architecture description languages

Lecture 8: Software Modeling and Design

Learning Objective: Learn about the principles of Object-oriented design and UML, and learn about the principles of automatic code generation. Students are expected to associate the principles with the examples, and be able to eventually apply these in practice.

- Object-oriented design and UML
  - Object Terminology
  - Object-oriented Design: Example
  - Why model with UML?
  - UML Diagrams
    - UML (Static) Structure Diagrams
    - UML Behavior Diagrams
  - UML for Embedded Systems
  - Key Attributes of UML that are Important to Embedded Systems
  - System Design Methodology using UML
  - UML and Platform-based Design
  - UML for Embedded Systems: Examples
- Automatic code generation from models
  - Approaches for Automatic Code Generation from Object Models
    - Structural, behavioral, translative code generation
  - Code Generation from UML Models
    - Translative Code Generation from UML Models: Example
  - Code Generation using Real-time Workshop (Simulink Coder)
    - Simulink Coder: Key Features
    - Using Simulink Coder
      - Working with Targets
      - Working with Data
      - Generating Code
      - Executing Code in a Real-Time Environment
      - Tuning Parameters and Logging Data
    - Simulink Coder: Example

Lecture 9: System and Architecture Modeling

Learning Objective: This lecture is devoted to SysML, a graphical modeling language for systems engineering by the Object Management Group (OMG) and the International Council on Systems Engineering (INCOSE), developed in response to the UML used in software engineering. SysML supports the specification, analysis, design, verification and validation of systems that include hardware, software and data and is relevant to embedded systems modeling. It provides semantics (meaning) and notation to embedded system architecture.

- Learning Objectives of Module III: Modeling
- Understand System modeling techniques
- Heterogeneous system modeling
- SysML diagrams and language concepts
- How to apply SysML as part of Platform-based Design Process

Lecture 10: Real time operating systems modeling

Learning Objective: In this lecture, the students will learn about the basic concepts of real-time operating systems (RTOS), classic real-time task models, basic real-time scheduling algorithms and how to perform schedulability analysis for both static- (e.g., rate monotonic (RM)) and dynamic- (e.g., earliest deadline first (EDF)) priority scheduling. The student will exercise these concepts through both homework and programming assignments. We will start with assignments on RTOS concepts, basic real-time task models, and basic scheduling algorithms. Ask the student to manually calculate various performance metrics based on different real-time task sets and evaluate their schedulability under different scheduling policies. At the end of the lecture, the students will be assigned a programming assignment and asked to design and implement a customized real-time scheduler using VxWorks.

- Basic concepts of RTOS and why is it different from a traditional OS?
- RTOS fundamentals and various real-time task models o Soft real-time and hard real-time systems
    - Non-preemptive vs. preemptive task models
    - Periodic, sporadic and aperiodic task models.
- Real-time scheduling concepts and basic scheduling algorithms o Feasibility and schedulability
    - Metrics of scheduling algorithms
    - Monotonicity of scheduling algorithms.
- Schedulability analysis for both static- and dynamic-priority scheduling o Schedulability analysis for Rate Monotonic scheduling (RM)
    - Schedulability analysis for Deadline Monotonic scheduling (DM)
    - Optimality among static-priority algorithms
    - Utilization-based schedulability test for RM
    - Utilization-based schedulability test for EDF
    - Comparison between RM and EDF
- Examples in VxWorks

Lecture 11: Industry Case Studies

Lecture 12-14: Distributed and networked systems modeling

Learning Objective: The objective of this part of the lecture is to cover basic concepts that are required for designing reliable distributed systems. By the end of these three lectures, students will understand the critical theoretical aspects of designing reliable distributed systems and are able to consider them while designing/building such systems. Such understanding will help designers to build systems that avoid corner case design flaws (e.g., infinite loops, system halt due to broken communication) in practice.

- The Challenge of Time Synchronization Using Physical Clock
  - Definition of global state, algorithms for synchronization using physical clock
  - The concept of logical clock
  - Concept of global snapshot in a distributed system
- Design of Reliable Multicast Communication Protocols
  - Design of Reliable Multicast Algorithms, Concepts of Message ordering in distributed communications
- Design of Leader Election Protocols
- Design of Protocols for Mutual Exclusion
- Brewer's Conjecture
- Algorithm for Practical Byzantine Fault Tolerance
- Transactions and Concurrency Control
- Deadlock

Lecture 14: Course Review

USEFUL READING.

Key References

1. Balarin, F., M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, C. Passerone et al. 1997. Hardware-Software Co-Design of Embedded Systems. Boston: Kluwer.
2. Bouyssounse, B., and J. Sifakis. 2005. Embedded Systems Design: The ARTIST Roadmap for Research and Development. New York: Springer.
3. Electrical Engineering, Berkeley Department of, and Computer Science. 2012. 'EECS 249 Design of Embedded Systems'. https://chess.eecs.berkeley.edu/design/.
4. Giese, H., G. Karsai, E. Lee, B. Rumpe and B. Schätz. 2007. Model-Based Engineering of Embedded Real-Time Systems. Springer.
5. Lee, E.A., and S. A. Seshia. 2017. Introduction to Embedded Systems, Second Edition.
6. Sangiovanni-Vincentelli, A., 2007, Quo vadis, SLD? reasoning about the trends and challenges of system level design, Proceedings of the IEEE 95 (3), 467-506

7. http://retis.sssup.it/~marco/teaching/embeddedsystems/lessons/

Copyright. Copyrighted materials within the course are only for the use of students enrolled in the course for purposes associated with this course and may not be retained or further disseminated.

Grading. Student grades will be based upon: Homework 70%, In-Class Communication: 10%, Final Project Report: 20%.

Due Dates and Late Policy. All due dates will be identified in blackboard when the work is posted. Deadlines are based on Eastern Standard Time; if you are in a different time zone, please adjust your submittal times accordingly. The instructor reserves the right to change dates accordingly as the semester progresses. All changes will be communicated in an appropriate manner.

Homework. Homework assignments will be posted on HuskyCT. Homework assignment due dates will be given with the assignment. NO late homework will be accepted as the homework will often be discussed in class. Each problem will be graded on a scale of 0-100. Homework should be prepared as follows: Introduction (10% of grade): Problem Statement and Solution Strategy, Theory (20% of grade): Assumptions and Model Equations, Solution (30% of grade): Plots and/or Tables with the Results, Discussion (20% of grade): Observations, Comments, Conclusions, Appendix (20% of grade): Commented Code (if appropriate)

Project, Presentations and Project Report. A project is to be developed by student groups, which is expected to evolve during the entirety of the track. The portion of the project that is to be executed in this course refers mainly to design project identification, challenge quantification, significance and relevance to the MBD philosophy and plan of attack. The final deliverable (presentation) should identify all the aforementioned elements in a quantifiable manner and suggest a strategy for solution.

Student Conduct. http://www.dosa.uconn.edu/student_code.html. Students are responsible for adherence to the University of Connecticut student code of conduct. Pay attention to the section on Student Academic Misconduct, "Academic misconduct is dishonest or unethical academic behavior that includes, but is not limited, to misrepresenting mastery in an academic area (e.g., cheating), intentionally or knowingly failing to properly credit information, research or ideas to their rightful originators or representing such information, research or ideas as your own (e.g., plagiarism)." Examples of academic misconduct in this class include, but are not limited to: copying solutions from the solutions manual, using solutions from students who have taken this course in previous years, copying your friend's homework, looking at another

student's paper during an exam, lying to the professor or TA and incorrectly filling out the student workbook.

Attendance. Students should make every effort to attend the live sessions and to talk with students in the Slack chat forum to get help and assistance from others. It is practically impossible to follow the class if classes are missed.

Absences. Make-up of missed exams requires permission from the Dean of Students, see "Academic Regulations." Midterm-exams are treated the same as Final Examinations. Students involved in official University activities that conflict with class time must inform the instructor in writing prior to the anticipated absence and take the initiative to make up missed work in a timely fashion. In addition, students who will miss class for a religious observance must "inform their instructor in writing within the first three weeks of the semester, and prior to the anticipated absence, and should take the initiative to work out with the instructor a schedule for making up missed work."

Adding or Dropping a Course. If you should decide to add or drop a course, there are official procedures to follow:

- Matriculated students should add or drop a course through the Student Administration System.
- Non-degree students should refer to Non-Degree Add/Drop Information located on the registrar's website.

You must officially drop a course to avoid receiving an "F" on your permanent transcript. Simply discontinuing class or informing the instructor you want to drop does not constitute an official drop of the course. For more information, refer to the online Graduate Catalog,

Academic Calendar. The University's Academic Calendar contains important semester dates.

Students with Disabilities. Students needing special accommodations should work with the University's Center for Students with Disabilities (CSD). You may contact CSD by calling (860) 486-2020 or by emailing csd@uconn.edu. If your request for accommodation is approved, CSD will send an accommodation letter directly to your instructor(s) so that special arrangements can be made. (Note: Student requests for accommodation must be filed each semester.)

Course Schedule*

* Schedule is tentative and may change
[1] First Date indicates release of lecture modules

Instructor's Contact Information:

- Fei Miao: fei.miao@uconn.edu; Phone: (860)486-3471
- Office Hours: TBD

Helpful Links:

- Virtual Computer Lab at UConn: http://skybox.uconn.edu/
- Course Material: https://lms.uconn.edu
- Institute for Advanced Systems Engineering: http://www.utc-iase.uconn.edu/