Cyber-Physical Design and Analysis

Instructor

Eric Feron, School of Aerospace Engineering

Goal

Relation to other courses, programs and curricula: Cyber Physical Systems (CPS), including examples from Automotive, Medical, and Aerospace applications. Design methods: techniques of control, trajectory and activity design and planning; humans and CPS. Formalisms: Differential equations, finite-state machines. Analysis: Correctness, safety, liveness, simulation and formal methods. Design processes and tools: Requirements engineering: Vee diagrams, Model-Based Systems Engineering, System architectures, automated code generation. Reference documents and regulations for CPS.

Readings

Readings will be drawn from survey papers, research papers and published standards.

Grading

- Class participation as measured on Piazza: 5%
- Homework: 40%
- Projects: 40%
- Final: 15%

Prerequisites

Reasonable familiarity with programming and willingness to learn new languages. CS6310 ”Software analysis and design” is helpful.

Learning objectives and educational outcomes

Students will gain working knowledge, sufficient to:

1. Apprehend the techniques of software-based management of physical systems.
2. Become familiar with the notion of feedback.
3. Learn techniques for system control and guidance.
4. Appreciate the role of humans in CPS.
5. Learn about formal methods for CPS analysis.
6. Understand industrial CPS development practices, and available computer-aided design tools for system and architecture design, Model-Based Systems Engineering (MBSE), and software generation.
7. Understand the regulatory framework for CPS and related publications.

Academic integrity

Students are encouraged to discuss the problem sets and readings outside of class. However, everyone must submit their own work, and you may not share code or answers. If your discussion with another student helped you make a breakthrough on a difficult problem, that is fine, but give credit in your HW. The projects may be performed in teams. Suspected cases of honor code violations will be handled through the Office of Student Integrity. If you have a question about collaboration policy, please ask.

Learning Accommodations

If you have any accommodations you need to inform us during the first week of classes, and provide us with the detailed accommodation approval letter from the GT Office of Disability Services. We need to confirm during the first week of classes (by the first Friday at 4pm) that we can accommodate your requests. If you dont get approval from us by the first Friday at 4pm then we cannot accommodate your requests.

Excused Absences Policy

`http://www.catalog.gatech.edu/rules/4/`

Plagiarism

Plagiarism is a violation of the GT honor code. Your homeworks and projects will be checked with auto-checkers to detect plagiarism. All violations will be reported to the GT Office of Student Integrity, and you will be given a 0 on that component of the grade (projects or homeworks) and your course grade will be dropped one letter grade (OSI may impose stricter penalties, especially if you have prior offenses). It is a lot easier to inadvertently plagiarize than you might think. Think twice before using cut/paste tools, even if the source is your own works.

Cheating

Any incidents of suspected cheating will be treated in the same manner as plagiarism (zero on that component + 1 course letter grade lower) and will be reported to theGT Office of Student Integrity.

Topic outline

1. CPS introduction
   (a) Course introduction
   (b) CPS definitions
   (c) CPS trends
   (d) Areas of interest
   (e) Fundamental approach
   (f) CPS examples
   (g) CPS Genesis, Modeling, Design, Verification and Validation, Assembly and Deployment
2. Review
   (a) Calculus
   (b) Differential equations
   (c) Markov models
   (d) Linear systems
3. Models
   (a) Nature and Computation Myths: Airborne Collision avoidance examples
   (b) From Continuous to discrete dynamics
   (c) Other CPS representations

   (d) Examples: Water tank, spring oscillator
   (e) Communication systems
   (f) Modeling trade-offs: Faithfulness, manageability, level-of-detail, accuracy

4. Low level CPS design
   (a) Canonical Example: Stopping a car
   (b) Feedback
   (c) Reduced-gravity Drone
   (d) Linear Control
   (e) Controllability: Train example

5. Mid-level CPS design requirements
   (a) Trajectory Planning and examples
   (b) Aviation example
   (c) Typical requirements

6. Mid-level CPS design
   (a) Guidance techniques
   (b) Classical optimization and examples
   (c) Dynamic Programs
   (d) Automotive example
   (e) Rapidly-exploring Random Trees

7. High level CPS design
   (a) High-level decision-making: Linear Temporal Logic (LTL)
   (b) Self-driving car example
   (c) LTL and trajectory planning
   (d) Reduced-gravity drone example

8. Low-level humans and CPS interactions
   (a) Humans and CPS
   (b) Human control and workload
   (c) Handling qualities
   (d) Automotive example
   (e) Caravan-trailer sway alleviation

9. High-level human and CPS interactions: Aerospace perspective

   (a) Aviation context
   (b) Human-automation interaction
   (c) Allocating automation between humans and computers
   (d) Flexibility issues
   (e) Autonomy and complexity
   (f) Automation as a state machine
   (g) State machine observability by humans

10. Hardware-software co-design
    (a) Mechanics and software
    (b) Jetpacks!
    (c) Hardware vs. Software
    (d) Fast electronics
    (e) Accounting for physics

11. Processors and Sensors
    (a) Sensors and CPS - trends
    (b) Sensors, CPS, and IoT
    (c) Actuators and servos
    (d) Embedded CPS architectures
    (e) Communications
    (f) Processors

12. Systems Engineering, general approach
    (a) Vee Design cycle and activities
    (b) Industry standards and advisory documents
    (c) Tools and frameworks: Software-centric viewpoint
    (d) Model-based engineering pitfalls
    (e) AADL virtual integration cost savings

13. Architecture Analysis and Design Language (AADL): General Principles and positioning
    (a) Why modeling in design?
    (b) Models, Processes, and tools
    (c) AADL introduction
    (d) AADL Components: software, hardware
    (e) AADL properties

14. AADL Crazyflie Case Study
    (a) Modeling
    (b) Implementing a UAV control logic
    (c) AADL functional chain
    (d) Flow analysis
    (e) Latency / real-time scheduling
    (f) AADL, middleware, and code generation
    (g) Error modeling and analysis
15. Formal methods
    (a) Concerns of formal methods
    (b) Concerns about formal methods
    (c) Abstractions
    (d) Abstract interpretation
    (e) Model Checking
    (f) Hunting for invariants
16. Future of CPS