AIAA 2013-4809

Guidance, Navigation, and Control and Co-located Conferences
August 19-22, 2013, Boston, MA
AIAA Infotech@Aerospace (I@A) Conference

Aerospace Cyber-Physical Systems Education

Ella M. Atkins[*] and Justin M. Bradley[†]
Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, 48109

Modern Aerospace systems are cyber-physical, comprised of physical components but commanded and controlled by "cyber" (computing and communication) elements. It is widely acknowledged that co-design across cyber and physical elements will provide a better-performing holistic system, but few of today's engineers have adequate preparation to model, optimize, and simulate both. Even working in teams, "language barriers" are problematic – most Aerospace engineers are not comfortable with object-oriented abstractions and model-based real-time system design principles from computer science. Conversely, computer scientists have little expertise in the linear and nonlinear calculus-based models describing the motion of complex air and space vehicles, not to mention associated algorithms ensuring stable envelope-wide guidance and control. This paper describes the cyber-physical system (CPS) from both perspectives, and then describes grand challenges in research and applications associated with the CPS. Focus is then turned to education. A review of the traditional Aerospace and computer science curricula is provided first, along with a summary of feedback received from the Aerospace industry regarding knowledge and skill needs for new-hires. A review of CPS courses introduced in other programs is provided, comparing and contrasting the different course perspectives on what is "fundamental" to the CPS. Strategies to migrate content of such courses to the undergraduate Aerospace curriculum are discussed.

I. Introduction

Modern Aerospace systems have a tight coupling between onboard cyber systems (e.g. processing, communication) and physical elements (e.g. platform structure, sensing, actuation, and environment). Researchers have begun to explore and exploit "cyber-physical systems" or CPS, defined as "engineered systems that are built from and depend upon the synergy of computational and physical components."[1] Within Aerospace, the flight dynamics and control community has traditionally tackled challenges most closely related to CPS, with bias toward the "physical" vehicle-centric aspects of the CPS. Conversely, the computer science community has focused on managing the "computational" side of CPS. In recent years, computer science researchers have integrated concepts from control theory into the optimization and online regulation of real-time computing resources. Control systems researchers have also incorporated ideas from computing theory, relying on concepts such as finite state machines and timed automata to model process state. Control theorists also use models of logic coupled with models of dynamics to form hybrid systems.[2] Existing models for digital control and time-delay systems have experienced resurgence in this age of co-optimization over physical and computational resources, and a myriad of domain applications from Aerospace and other fields have tangibly benefitted from the coupling of cyber and physical models. Modern curricula, however, have remained largely stove-piped. Aerospace students still see little required course content in computer science beyond a single "freshman programming" course, while computer science students do not explore calculus-based models of physical systems once their basic Calculus and Physics course sequence has been completed. Importance of legacy curriculum content coupled with credit limits challenge the introduction of new material in cyber-physical systems. While emerging design/build/test courses show some promise in helping introduce at least the basic concepts of CPS, theoretical underpinnings are difficult to introduce in this context.

Aerospace Engineering undergraduate coursework focuses on traditional branches of Aerospace including structures, gas dynamics, and flight dynamics and control. With increasing focus in industry placed on complex avionics and software for modern spacecraft, commercial aircraft, and unmanned aircraft systems (UAS), understanding the coupling between cyber and physical resources is important in the curriculum to arm our graduates with at least a fundamental understanding of CPS principles.[3] While Aerospace education has incorporated modern problem-based learning,[4] the focus is generally on vehicle aerodynamic, structural, propulsion,

[*] Associate Professor, Aerospace Engineering Dept., University of Michigan, Ann Arbor, MI, Associate Fellow.
[†] Ph.D. Candidate, Aerospace Engineering Dept., University of Michigan, Ann Arbor, MI 48109, Student Member.

and control actuation properties, not on the interplays between vehicle physical design and operation with the avionics and software design and operation.

This paper presents a review of Cyber-Physical Systems, from both the control-theoretic and computer science perspectives. Grand challenges identified for CPS are introduced across domains, providing perspectives on where the community is focusing current attention. Current curricula are then examined, both in Aerospace and computer science undergraduate programs, followed by a discussion of feedback from the Aerospace industry on how we need to improve education of undergraduates with respect to their “computing knowledge”. A review of CPS courses introduced in other programs is provided, comparing and contrasting the computer science and control-theoretic course perspectives on what is “fundamental” to a CPS. Options are presented for introducing CPS to Aerospace students at the undergraduate and graduate levels, building on background students possess from existing courses.

II. Cyber-Physical Systems Background

The communities involved with the emerging field of cyber-physical systems agree that research in this area must focus on consideration of a computing system acquiring data from and making decisions in a complex physical environment. However, the nature of data, translation and abstraction of this data, and subsequent control decisions made in CPS depend substantially on both the application and the perspective taken by the research group studying CPS. For example, CPS have been proposed for domains ranging from medical monitoring[5] to automotive (engine control)[6] applications. CPS decisions range from coordination[7] and fault-tolerant control[8] of physical actuation decisions to ensuring the security of information[9] acquired from the network/cloud. In summary, CPS are inherently multi-disciplinary, challenging educators and practitioners to maintain knowledge of fundamental and application concepts that cut across as a minimum the fields of dynamics and control (control theory) and computer science (real-time computing). Below are summaries of the perspectives to CPS taken by each of these two groups. As will be discussed, there are many common aspects to the CPS that cut across all perspectives, yet the emphases taken in modeling and decision-making still tends to be quite discipline-specific, leading to further challenges in developing a CPS curriculum that is sufficiently perspective-independent to enable those educated in CPS from one perspective to successfully interact with those educated with respect to another viewpoint.

A. Dynamics and Control Perspective

The field of dynamics and control has significant breadth and depth. Historically, system dynamics models have been specified with continuous-time differential equations. For mobile vehicles, these models typically specify the rigid-body motion of a body coordinate frame with respect to an inertial reference frame. Robotic or flexible-structure vehicles may require additional models to account for non-rigid degrees of freedom. Control theory is an interdisciplinary branch of engineering and mathematics that deals with complex physical systems. A feedback controller manipulates the inputs to the system to obtain the desired behavior through commanding outputs to physical actuators of the system.[10] The classically-trained control theorist therefore preferentially models dynamic systems using differential equations with the goals of understanding system response and defining feedback control laws that can be mathematically proven optimal and [asymptotically] stable. The control theory community has investigated and adapted modeling and computational methods originally derived for purposes outside control theory, e.g., finite state automata from computer science and neural networks from biology. Neural networks, a concept originating from study of the human nervous system, have been shown a computationally-efficient method for adaptive feedback control.[11] Finite state machines have been adapted by control theorists to compactly represent mode-switched control systems, a formulation known as hybrid systems.[12]

The hybrid system formulation, combining continuous-valued physical state and discrete-valued mode or state into a single formulation, has shown promise for integrating the discrete state models used for typical computing systems with continuous state models used to model physical systems. Proof that the continuous state of a hybrid system will be asymptotically stable across all possible state transition “jumps” as well as within each state is difficult, calling upon many of the more advanced concepts covered in graduate control theory courses (e.g., Lyapunov stability theory). For this reason, computer scientists have not embraced hybrid systems models for holistic CPS modeling. Additionally, because of the challenges associated with proving stability, practical hybrid systems models tend to be minimalist with respect to complexity of the discrete state-space. Most hybrid automata models therefore use modes only to distinguish different continuous-state dynamics (e.g., linearized about different equilibrium points), not to represent the complete memory of information state. For this reason, control theory students who learn about state machines think of the "discrete states" in terms of switching between continuous physics-based models.

Another example illustrating the control-theoretic perspective is in the representation and manipulation of graphs. Graph theory has become popular in the dynamics and control community for activities such as cooperative control of multiple vehicles.[13] The basic concept of a graph as a collection of nodes (vertices) and edges, directed or undirected, is universal. Properties such as whether the graph is cyclic, a tree, etc. are also universal. Analysis, manipulation, and use of the graph as a modeling tool quickly become community-specific. For the cited formation control application,[13] analysis tools focus on vehicle and formation stability, e.g., using a Nyquist criterion. This emphasis is appropriate as the application is to manage the physical motion of a collection of vehicles. Because graphs representing realistic formations of vehicles would be relatively small in size, little consideration is given to how a [large or dynamic] graph topology might be represented and managed. The implicit assumption is that whatever graph is built can be managed and manipulated on computing and communication systems in real-time. From a pedagogical viewpoint, such an approach to teaching graph theory gives students substantial insight into how a formal analysis of physical system stability and performance might be conducted, but it gives students little insight or even appreciation for the memory, computation, and communication challenges facing the computer system that must manage very large, dynamically-evolving graphs.

B. Real-time Computing Perspective

The field of real-time computing is much newer than is dynamics and control, given that computers were only scaled to a size that could be embedded in physically-mobile systems a few decades ago. Computing technology and software engineering have, however, progressed extremely rapidly, as have communication networks. Real-time computing (RTC), or reactive computing, is the study of hardware and software systems that are subject to "real-time constraints", e.g. response times measured from initial event (task arrival) to system response (task completion). Real-time programs must guarantee response within strict time constraints, often referred to as "deadlines".[14] Real-time communication networks analogously must guarantee delivery of data within time constraints. Researchers in real-time computing build discrete models in which computational tasks or communication packets must be allocated to and scheduled on available resources. Computing theory forms the basis for the formal specification and characterization of static and dynamic data structures, computing algorithms, and communication protocols to which real-time computing analyses are applied. Computational complexity analysis provides worst-case (upper) bounds on algorithm computational time and memory requirements enabling classical scheduling algorithms such as rate monotonic[15] to guarantee hard real-time execution deadlines can be met. Task and message packet models are inherently discrete, placed on integer or real-valued timelines for each resource.

Load management and balancing algorithms have adapted principles from feedback control to RTC purposes, e.g., feedback scheduling for real-time computing networks.[16,17] In large-scale networks, the set of computing and communication tasks to schedule, many with precedence and exclusion constraints, can be quite large. Large-scale scheduling applications have therefore encouraged RTC researchers to develop approximate algorithms and hierarchical network structures that are scalable. The emphasis on accommodating large-scale problem spaces with necessary compromise in performance, including degradation of tasks as necessary per protocols such as Quality of Service (QoS),[18] has resulted in an emphasis on scalability for RTC systems, as opposed to the emphasis on guaranteed stability for physical vehicle control systems. Both are appropriate emphases, but these differences also result in perspective differences between control theory and RTC communities.

In the classroom, the RTC student will be exposed to models of computation, including data structures, algorithms, and complexity analysis. In the context of the CPS, the RTC student will therefore define problems in terms of data to be packaged, transferred, processed, and stored, and the algorithms by which the associated computing and communication tasks are achieved. RTC students therefore devote substantial time to understanding the application software, the middleware, and the communication protocols by which these activities are accomplished. With this emphasis assignments often involve writing and testing code. While computing networks may be represented by the same type of universal graph as is used to represent connectivity between vehicles in a formation, the number of nodes and edges of a computing network graph may be many orders of magnitude larger than counterparts representing a vehicle formation network. The network may also be highly-dynamic, with nodes going in and out of range of specific network hubs/routers at a rapid pace. For this reason, emphasis in RTC is placed on efficiently representing and managing dynamic network data structures. Model and code complexity has also challenged formal validation and verification methods; while covered in courses, the RTC community necessarily combines practices such as nightly builds and tests with formal methods that are limited in their ability to exhaustively model and verify complex codes.

III. CPS Grand Challenges

CPS models ideally integrate models of physical vehicles, their environment, and associated human users, operators, collaborators, or bystanders to yield efficient, high-confidence system-of-systems. CPS are therefore complex but must meet metrics associated with reliability, robustness, predictability, security, etc. It is acknowledged that current approaches to system design, development, certification, and operation typically separate design and analysis of physical subsystems (e.g., electromechanical) from the designs of the software, command, and control systems. As we enter a time where society is eagerly anticipating the approval of autonomous systems such as driverless cars, it is important to recognize and appreciate some of the grand challenges remaining for the research and application communities.

Several researchers have articulated CPS grand challenges worthy of consideration as grounding for a CPS-oriented curriculum. Poovendran[19] defines a vision in which "networking is employed at multiple and extreme scales, complexity lies at multiple temporal and spatial scales, dynamics exist in system reorganization and reconfiguration, high degrees of automation and control loops close at all implemented scales, system longevity ranges from years to decades (e.g., buildings, aircraft), and extreme heterogeneity is seen across devices and protocols." For transportation, phrases such as accident-free, zero-emissions, self-certified, zero-defects/recalls are cited as grand challenges. Sha et al[20] indicate that two of our greatest challenges are global warming coupled with energy shortage and providing services for the rapidly-aging population. CPS cited to assist in facing these challenges include "zero-energy" buildings that tightly integrate networked sensors and computational elements with physical building elements, including management of renewable energy sources. CPS-enabled systems such as service robots and driverless cars can assist the elderly, reducing the increasing demand for nursing home care while improving quality of life. Rajkumar et al[21] offer a vision of CPS that, if achieved, will have broad impact in scientific, economic, and technology sectors. CPS grand challenges cited include blackout-free electricity generation, safe and rapid emergency evacuation, and reduced testing and integration time for complex safety-critical complex CPS systems such as avionics.

In the Aerospace sector, both aviation and space application domains can benefit substantially from advances in CPS. As summarized by Long,[22] the Aerospace industry has incurred tremendous cost overruns and delivery delays due to issues in software development and testing. A number of mission failures can also be traced back to software issues. Space systems ranging from robotic explorers to smart launch vehicles have relied on software control systems since the advent of the space age. New missions with more demanding requirements with respect to complexity, long-duration autonomous operation, cost reduction, and data throughput further challenge CPS, particularly as new operators such as those launching CubeSats enter the field.[23] Aviation faces analogous certification, trust, and acceptance challenges to those being experienced by advocates for driverless cars. Different barriers exist, however. In aviation, psychology (fear of flight) may limit automation for commercial transport, although CPS support for human flight crews can improve over time. Perhaps the greatest challenge for aviation is integration of unmanned aircraft system (UAS) into the National Airspace System (NAS), for which regulations were designed during a time of human-piloted flight. The CPS grand challenges for UAS are therefore twofold: continued pursuit of safer, more efficient UAS, and safe integration with legacy, manually-piloted systems – with ultimate goal of guaranteeing "collision-free flight" with respect to other aircraft in the sky and safety for people and property on the ground.

IV. Existing Undergraduate Curricula and Feedback from the Aerospace Industry

Before discussing how CPS might be introduced to Aerospace students, we first ground the discussion by reviewing required undergraduate coursework in two majors: Aerospace Engineering and Computer Science and Engineering. Although the following discussions focus on curricula found at the University of Michigan, the authors investigated curricula at other major universities and identified similar course sequences. One notable exception is the Aeronautics & Astronautics program at MIT, where undergraduate students are exposed to substantially more content in information systems than in other University Aerospace programs. While the term "information system" is not interchangeable with "cyber-physical system" such interdisciplinary content definitely provides useful exposure to cross-cutting fundamentals.

The required core curriculum at the University of Michigan is common for all College of Engineering students. In addition to humanities and social sciences, students take four math courses (Calc I – Diff Eq), two Physics courses with labs, one Chemistry course with lab, and two college-wide engineering courses, one introducing students to project-based engineering (Engr100) and one four-credit introduction to computers and programming course (Engr101).[‡] While Engr101, taught multiple times by the first author, introduces freshmen to computational thinking in the context of C++ and Matlab, it is an "island", i.e., most students enter with no programming background and then take no further courses in this area. Non-EECS (electrical engineering and computer science) majors see little further content in "computational thinking" in their undergraduate coursework. Engr101 therefore has no reinforcement in most majors thus has limited impact on a student's preparedness for more advance computing concepts such as those needed for the RTC branch of CPS.

A. Aerospace Engineering (Aero)

The required undergraduate Aerospace curriculum at the University of Michigan is structured around three pillars: structures, gas dynamics, and dynamics and control. Students are not offered tracks – many faculty contend that students are not yet prepared to decide what Aerospace path they want to take thus a unified curriculum focusing on the "fundamentals" is most appropriate. Each of the three pillars has a sequence of required courses that build to senior capstone design and laboratory courses. In structures, students take materials science and a sequence of two structures courses. In gas dynamics, students take a series of two courses in aerodynamics plus a propulsion course. In dynamics and control, students take a sophomore dynamics course plus an introductory Aerospace course that covers atmosphere models, coordinate systems and kinematics, steady flight, and systems theory. Students then take a space flight mechanics and aircraft dynamics and control course sequence. A sequence of two lab courses, a seminar course, and a capstone design course (aircraft and space systems options are available) round out the required course list. In terms of software, Matlab and Simulink are the most widely-used software packages. Their use, however, focuses almost exclusively on promoting understanding of concepts related to calculus-based modeling and simulation, using built-in functions that "black box" most all aspects of how data is represented, processed, and communicated.

Two Aero courses have recently shown some promise to expand student exposure to "computational thinking". The introductory Aerospace course providing coverage of steady flight principles has evolved to introduce students to fundamentals of logic and state machines. This extension provides early insight for Aerospace students into the "control theory" branch of CPS. Another course, to be first required in 2014, will provide an early design-build-test experience for undergraduate sophomores. Although primarily serving the need for more hands-on education in the curriculum, this course will provide an embedded programming experience for students, including fundamental RTC concepts in periodic data acquisition and real-time processing and communication with limited resources. To some extent, these two courses will better prepare Aerospace sophomores for a more in-depth education in CPS. The challenge then is to follow-up with CPS exposure in the junior, senior, and graduate years without continuing to require that students with strong CPS interest double-major in computer science.

B. Computer Science & Engineering (CSE)

In CSE, undergraduate sophomores take discrete mathematics and statistics courses. A sequence of two courses introduces sophomores to object-oriented programming, data structures, and algorithms. Foundations of computer science and computer architecture courses plus a course in technical communication round out the program core requirements. CSE offers more options or tracks to its undergraduates than Aero, which follows a single track except for the senior capstone course. CSE students are required to take a major design experience course from an approved list. Students also must take 26 technical elective credits, of which 16 must be upper-level CSE courses. Students can choose from a wide variety of upper-level courses ranging from artificial intelligence and compiler construction to operating systems, formal verification, robotics, and user interfaces.

The flexibility in the CSE program provides an opportunity for students interested in CPS to pursue a related course sequence within the major, enabling broad or deep coverage in related topics such as real-time computing (RTC). While technical electives offer CSE students the option of broadening their CPS-related expertise to control theory or more generally calculus-based modeling courses, upper-level students who are not double-majoring and

[‡] An accelerated introduction to programming option, Engr151, was introduced by the first author to help students with programming background find a more appropriate challenge. Few students electing Engr151 pursue Aerospace, however.

who have focused most of their attention on discrete models, algorithms, and data structures, choose not to pursue the more advanced courses in physics-based modeling.

C. Feedback from the Aerospace Industry

Academic units typically invite industrial advisory boards to attend annual program reviews and provide recommendations for the department. Feedback is broad but often focused on curriculum issues since industry has a strong interest in recruiting graduates who are suitable for employment at their companies. The first author has some experience with advisory board feedback at two universities. Each year, the advisory board discusses the need for improved “computing” education. The problem is that, for Aerospace, most board members have little personal expertise in what has been defined here as “CPS”, so most think of “computing” in terms relevant to their Aerospace sub discipline. For example, structures experts consider CAD (computer-aided design), FEM (finite element modeling), and CAM (computer-aided manufacturing) software packages to be critical exposures for undergraduate students. While these packages provide significant insight into the design and manufacturing of Aerospace structures, they do not promote the kind of fundamental understanding of computational thinking[25] essential to prepare students for further study in CPS.

Some members of industrial advisory boards, particularly those who have managed aircraft or spacecraft programs that have faced significant software and avionics cost overruns, have expressed support for further student exposure to software development and testing. Of particular interest to the advisory board, and to many students, is exposure to embedded system programming and/or software engineering. The challenge then is to fit such coursework into the Aerospace curriculum. During the first decade of her time in academia, the first author idealistically embraced support from the advisory board, believing such support would lead to curriculum change. Unfortunately, the realities of credit limits and lack of recognition by other faculty of the importance of ‘computational thinking’ (much less CPS) in the required curriculum have resulted in few changes. In fact, there has been some decrease in coverage of software engineering and embedded systems concepts in some Aerospace programs due to “black box” applications and off-the-shelf products, providing reduced motivation for students (and in many cases instructors) to ensure students are challenged to learn about even basic computing or RTC concepts.

V. CPS in the Aerospace Curriculum

A. Review of Existing CPS Courses

To provide context for the CPS topics that might be introduced to Aerospace students, we first review CPS courses introduced primarily in EECS departments. In summary, several such courses were identified. Two types of courses are offered: courses that focus on introducing as much of the underlying theoretical concepts of CPS as possible (e.g., http://www.cs.colorado.edu/~srirams/classes/doku.php/foundations_of_cyber-physical_systems_fall_2012, http://www.seas.upenn.edu/~lee/10cis541/), and seminar courses that focus on CPS applications but that at least enumerate the basic concepts and fields that together comprise a holistic CPS (e.g., http://www.ics.uci.edu/~eli/courses/seminar-s10/seminar-s10.html). The goal of instructors teaching a fundamental CPS course is to expose students to the basic models of cyber (real-time computing) and control systems, placing emphasis on how discrete and continuous modeling methods have been integrated in recent work, e.g., the hybrid systems formulation used for control of switched or multi-mode systems.

A recent textbook by Lee and Seshia[26] attempting to cover at least the basic foundations of CPS has been published and is available online. This book is divided into three parts: CPS modeling, CPS design, and CPS analysis. The modeling section contains content on continuous dynamics, discrete dynamics, hybrid systems, [hierarchical] state machine composition, and concurrent computation. The design section contains content on embedded and parallel processing, input and output, and scheduling. The analysis part of the text contains chapters on logic, equivalence, reachability analysis and model checking. While this text has a slight bent toward RTC, it also has a fair amount of coverage of quantitative models and analysis methods relevant for control theorists.

While this book can provide a foundation for curriculum development, even the authors clearly state in the preface that the book cannot be complete due to the breadth and depth of what we require in a CPS capable of addressing the types of grand challenges summarized above. Those who have offered other courses in CPS agree. The best preparation we can offer, therefore, is to ensure students are capable of both “computational” and "calculus-based" thinking, supplemented and reinforced by the modeling, design, and analysis tools covered in the Lee and Seshia text such as logic, hybrid systems, processing architectures and real-time scheduling.

B. Steps to CPS for Aerospace Undergraduates

Although recommended highly by the authors, it is unrealistic to think that Aerospace programs with a well-established curriculum will quickly or substantially alter the required curriculum to accommodate new material in CPS. There are therefore four means by which Aerospace student exposure to holistic CPS fundamentals can be improved: (1) Students can either double-major in Aerospace and CSE or major in one with minor in the other, (2) Existing required courses can be altered or supplemented with material related to CPS, (3) Technical electives covering CPS can be promoted to students, and (4) One or maybe even two new required courses with substantive CPS content can be introduced (in the long-term).

Option (1) is viable for top students and is recommended by the first author as the clear option available today. It is our observation that students with substantive background (and good GPAs) in Aerospace and CSE are in high demand. Examples of option (2) are beginning to appear in Michigan's Aerospace curriculum; as discussed above, course content in embedded programming (thus CPS - RTC) and in state-space and system models (thus CPS – control) are beginning to be introduced at the sophomore level. The authors are still "working on" achieving significant impact with the third option. At Michigan, an Aerospace technical elective in "flight software systems" has been repeatedly offered, but with very low enrollments. Quite simply, because the CAD course is so popular and few technical elective courses are required, few students opt for the more "CPS-related" course. We are still working on option (4); to date the Aerospace faculty members have not as a group been supportive of eliminating existing required course credits in favor of any new more CPS-related course requirement(s).

VI. Conclusions and Future Work

This manuscript has outlined the emerging field of cyber-physical systems (CPS) with the goal of shedding light on educational perspectives and needs for the broader community then ultimately for Aerospace. Two viewpoints of the fundamental concepts of CPS are presented along with grand challenges and opportunities in the emerging field. To provide context for incorporating CPS into undergraduate curricula, current programs in Aerospace and Computer Science and Engineering at the University are overviewed. The Aerospace curriculum offers uniform and deep exposure to the calculus-based modeling and simulation concepts that form the three traditional "pillars" of structures, gas dynamics, and dynamics & control, but offers few technical and general elective credits as part of the required curriculum. As a result, most Aero graduates have little to no exposure to RTC fundamentals such as data structures, networks, or algorithm/code analysis (e.g., complexity theory). Conversely, the CSE program focuses on discrete models of logic, data representation, and algorithms, and offers substantial choice in upper-level courses that can meet program requirements. Most upper-level CSE students do not select courses that reinforce their early exposures to calculus-based models prevalent in control-theoretic CPS formulations.

While credit limits, the persistence of legacy curricula, and remaining insular attitudes challenge further progress in CPS education, students and practitioners must place substantial effort toward finding ways to expand their cross-cutting knowledge in CPS to advance state-of-the-art. With further recognition of the fundamentals of CPS and how they can be integrated and supported by engineers with interdisciplinary expertise, CPS research and education will certainly advance and grow to achieve the truly holistic concept those who coined the term "CPS" may have originally envisioned.

VII. Acknowledgements

The authors would like to thank Kang Shin and other CPS project collaborators for their input and guidance. This work was funded in part under NSF grant CNS-0930813.

VIII. References

[1] National Science Foundation (NSF), "Cyber-Physical Systems," NSF, [Online]. Available: http://www.nsf.gov/pubs/2013/nsf13502/nsf13502.htm. [Accessed 31 January 2013].

[2] Tomlin, C., Pappas, G., Sastry, S., "Conflict resolution for air traffic management: a study in multiagent hybrid systems," *Automatic Control, IEEE Transactions on* , vol.43, no.4, pp.509-521, Apr 1998 doi: 10.1109/9.664154.
[3] Ying, S., Venema, S., Corman, S.. Angus, I., and Sampigethaya, R., "Aerospace Cyber Physical Systems – Challenges in Commercial Aviation," *NIST Foundations for Innovation in Cyber-Physical Systems Workshop*, Chicago, IL, March 2012.
[4] Brodeur, D. R., Young, P. W., and Blair, K. B. "Problem-based learning in aerospace engineering education." In *Proceedings of the 2002 American Society for Engineering Education Annual Conference and Exposition,* pp. 16-19, Montreal, Canada, June 2002.
[5] Lee, I. and Sokolsky. O. "Medical cyber physical systems." *Design Automation Conference (DAC), 2010 47th ACM/IEEE*. IEEE, 2010.
[6] Jensen, J. C., Chang, D. H., and Lee, E. A., "A model-based design methodology for cyber-physical systems," In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International* (pp. 1666-1671). IEEE, July 2011.
[7] Kim, K. D. "Collision free autonomous ground traffic: a model predictive control approach," In *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems* (pp. 51-60). ACM, April 2013.
[8] Wang, X., Hovakimyan, N., and Sha, L. "L1Simplex: fault-tolerant control of cyber-physical systems," In *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems* (pp. 41-50). ACM, April 2013.
[9] Han, K., Potluri, S. D., and Shin, K. G. "On authentication in a connected vehicle: secure integration of mobile devices with vehicular networks," In *Proceedings of the ACM/IEEE 4th International Conference on Cyber-Physical Systems* (pp. 160-169). ACM, July 2013.
[10] Wikipedia, "Control Theory," http://en.wikipedia.org/wiki/Control_theory (last accessed Aug. 1, 2013).
[11] Sanner, R. M., and Slotine, J. J. "Gaussian networks for direct adaptive control," *Neural Networks, IEEE Transactions on*, *3*(6), 837-863, 1992.
[12] Alur, R., Courcoubetis, C., Henzinger, T. A., and Ho, P. H. *Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems* (pp. 209-229). Springer Berlin Heidelberg, 1993.
[13] Fax, J. A., and Murray, R. M. "Information flow and cooperative control of vehicle formations," *Automatic Control, IEEE Transactions on*, *49*(9), 1465-1476, 2004.
[14] Wikipedia, "Real-time Computing," http://en.wikipedia.org/wiki/Real-time_computing (last accessed Aug. 1, 2013).
[15] Liu, C. L., and Layland, J. W. "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, *20*(1), 46-61, 1973.
[16] Stankovic, J. A., He, T., Abdelzaher, T., Marley, M., Tao, G., Son, S., and Lu, C. "Feedback control scheduling in distributed real-time systems," In *Real-Time Systems Symposium, 2001.(RTSS 2001). Proceedings. 22nd IEEE* (pp. 59-70). IEEE, 2001.
[17] Krishna, C. M., and Shin, K. G., "On scheduling tasks with a quick recovery from failure," *Computers, IEEE Transactions on*, *100*(5), 448-455, 1986.
[18] Abdelzaher, T. F., Atkins, E. M., and Shin, K. G. "QoS negotiation in real-time systems and its application to automated flight control," *Computers, IEEE Transactions on*, *49*(11), 1170-1183, 2001.
[19] Poovendran, R., "Cyber–Physical Systems: Close Encounters Between Two Parallel Worlds [Point of View]," *Proceedings of the IEEE, 98*(8), 1363-1366, 2010.
[20] Sha, L., Gopalakrishnan, S., Liu, X., & Wang, Q. "Cyber-physical systems: A new frontier," In *Machine Learning in Cyber Trust* (pp. 3-13). Springer US, 2009.
[21] Rajkumar, R. R., Lee, I., Sha, L., & Stankovic, J, "Cyber-physical systems: the next computing revolution," In *Proceedings of the 47th Design Automation Conference* (pp. 731-736). ACM, June 2010.
[22] Long, L. N., "The Critical Need for Software Engineering Education," *CrossTalk*, *21*(1), 6-10, 2008.
[23] Klesh, A., Cutler, J., Atkins, E. "Cyber-Physical Challenges for Space Systems," *Proceedings of the International Conference on Cyber-Physical Systems (ICCPS*), IEEE, Beijing, China, April 2012.
[24] Bradley, J., Atkins, E. "Multi-Disciplinary Cyber-Physical System Optimization," *Proceedings of Infotech@Aerospace,* AIAA, June, 2012.
[25] Wing, J. M. "Computational thinking," *Communications of the ACM*, *49*(3), 33-35, 2006.
[26] Lee, E. A., & Seshia, S. A. *Introduction to embedded systems: a cyber-physical systems approach*. Lee & Seshia, 2011.