Chapter 10
Aerospace Engineering Curricular Expansion in Information Systems

Ella M. Atkins

Abstract This chapter investigates specific approaches to evolve the Aerospace Engineering curricula to increase coverage of the fundamentals of computer science and deepen student experience in programming. First, existing K-12, Aerospace Engineering, and Computer Science and Engineering curricula are examined. Multidisciplinary programs including robotics and Cyber-Physical Systems (CPS) are reviewed to provide insight into potential directions in which an Aerospace-centric program might expand. Student, faculty, and industry interests offer insight into key Computer Science and Engineering (CSE) content to infuse into next-generation Aerospace curricula. The approach being taken at the University of Michigan, the author's home institution, is described, including plans to increase curricular flexibility and introduce a new course providing students background in key computer science concepts such as data structures and complexity, computational science with application to Aerospace analysis and design, and embedded data management and control. A discussion of potential future curricular extensions into human–machine systems and electromechanical devices concludes the chapter.

10.1 Introduction

Modern Aerospace systems integrate traditional physical vehicle structural and mechanical components with avionics, software, and people. Embedded sensors and microcontrollers have substantially reduced aircraft weight and increased reliability through local data processing, redundancy, and lightweight communication. High-bandwidth sensor data streams and advanced decision algorithms require capable onboard computers. Computers with increasingly complex software are now tasked with managing the payload, controlling vehicle motions, and supporting increasingly autonomous decision systems. Manned vehicles require software and devices to assure an onboard pilot remains situationally aware whereas unmanned vehicles must rely on communication links to remote ground or mission control stations.

E. M. Atkins (✉)
University of Michigan, Ann Arbor, MI, USA
e-mail: ematkins@umich.edu

U. Durak et al. (eds.), *Advances in Aeronautical Informatics*,
https://doi.org/10.1007/978-3-319-75058-3_10

Operator interface designs require careful consideration of real-time data stream and communication link properties as well as human factors. A twenty-first-century Aerospace system designer faces the difficult challenge of understanding both the traditional physics-based models underlying structure, aerodynamic, propulsion, and flight dynamics foundations as well as understanding the computer science foundations necessary to design, implement, and validate/verify the avionics and software supporting onboard and network-wide data-to-decision systems.

To face this challenge, the industry has staffed Aerospace systems design teams with diverse technical experts organized to collectively provide the breadth and depth of expertise needed to produce modern Aerospace systems. For example, to design a fighter or transport aircraft, teams of Aerospace engineers propose and optimize aerostructural and vehicle control designs; teams of electrical engineering propose and optimize avionics and power systems designs; teams of computer scientists propose and optimize onboard, network, and off-board software; and teams of human interface experts, including pilots, propose logical user interfaces. Engine manufacturers organize into similar subteams to produce efficient FADEC (Full Authority Digital Engine Control)-equipped products that can be hosted on a variety of airframe designs. Specialized subteams make sense so long as the subsystem designs can be developed independently, with consideration of design elements exclusively within that field of study. For example, an Aerospace engineer offering Computational Fluid Dynamics (CFD) expertise will still have sufficient background in rigid body vehicle dynamics and structures from their undergraduate studies to communicate effectively with vehicle control and structural experts. Electrical engineers will have sufficient background in analog, digital, and power systems to optimize over all avionics system elements. Computer scientists will similarly have sufficient background in logic, data structures, algorithms, and software engineering practices to work as a team despite specializations ranging from real-time embedded computing to artificial intelligence and cybersecurity. Human factors or cognitive engineering teams understand operators and ensure interfaces are informative, intuitive, and not distracting/misleading. Human-centered designs assure the "people in the system", pilots, remote operators, passengers, the overflown public, remain situationally aware and are comfortable rather than threatened.

These four subteams form the four pillars supporting the modern "Aerospace System" enterprise as shown in Fig. 10.1: (Traditional) Aerospace Engineering, Electrical Engineering, Computer Science, and Human Factors and Operations. With reference to the USA education system, Aerospace system designers enter higher education programs with a K-12 background and first-year university experience that is nearly common across the engineering fields. This foundation provides excellent coverage of traditional mathematics through Calculus and the traditional sciences, e.g., physics and chemistry. Computer science, circuits, and human–machine system exposures tend to occur through extracurricular activities in robotics or game development, requiring universities to fill knowledge gaps in these areas. Following a nearly common year 1 curriculum, university students select one of the "pillars" or majors and specialize in that area for the remainder of their degree. Project-based experiences can expose students to the challenges related to other "pillars", but a

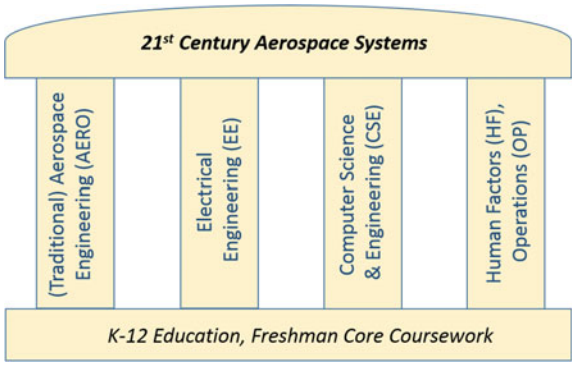

Fig. 10.1 Foundations of a Modern Aerospace System

minor or double major is required for a student to gain depth in any of these other pillar areas.

As Aerospace systems more tightly couple Fig. 10.1 "pillars" to achieve new levels of performance, it is clear new graduates will require a new level of crosscutting expertise. Today's entering college students increasingly recognize that future Aerospace Engineers will benefit from an improved foundation in computer science, while future Computer Scientists electing a career in Aerospace system design recognize they will benefit from improved foundation in vehicle aerostructural, electromechanical, and control systems. At the University of Michigan, we find evidence of the former in terms of the steadily increasing rates at which our Aerospace undergraduates pursue Computer Science minors or double majors. We find evidence of the latter in terms of the steadily increasing rates at which Computer Science undergraduates pursue robotics-centric coursework, research experiences, and student teams. In our growing robotics graduate program, entering students are eager to learn what they do not know. In particular, Aerospace and Mechanical Engineering graduates seek knowledge of software and autonomy, while Computer Science and Engineering graduates seek knowledge in electromechanical (robot) system design, build, and test. While robotics indeed provides multidisciplinary exposures, knowledge of the domain (e.g., Aerospace) tends to be limited, and graduates have statistically selected employment outside the Aerospace sector. It therefore remains critical for the Aerospace Systems community to work toward improved holistic education in Aerospace and Computer Science for the twenty-first-century workforce.

Researchers and educators have recognized the engineering/computer science curricular gap for more than a decade. The National Science Foundation (NSF) has established a robust Cyber-Physical Systems (CPS) program aimed at encouraging researchers in traditional engineering and computer science disciplines to closely collaborate on crosscutting research projects and to improve CPS education generally. The robotics community has also shared the vision of crosscutting expertise, placing importance on tight integration of sensors and mechanisms with capable perceive-decide-act logic and code. Independent programs dedicated to CPS and/or robotics, though still the exception rather than the rule, can offer guidance on what is possible in crosscutting curricular programs.

This paper proposes potential strategies for improving the exposure of Aerospace students to Computer Science foundations, and improving the exposure of Computer Science students to Aerospace/Robotics. Legacy curricula, degree credit limits, students' interests, and faculty expertise are all critical factors to examine in this context. Higher education curricula are already packed with required courses, so some sacrifice in traditional foundational background is required to maintain degree credit limits. Curricular additions must therefore be defined alongside methods to relax existing coursework requirements when needed. This paper is structured as follows. First, background in K-12 foundations, undergraduate Aerospace Engineering, and undergraduate Computer Science curricula is presented. Next, efforts to establish multidisciplinary programs in CPS and robotics are summarized, along with a review of open questions that must be addressed when considering options to infuse computer science into traditional engineering programs. Specific initiatives to improve computer science coverage in the University of Michigan's Aerospace Engineering curriculum are described in the context of student, industrial advisory board, and faculty input. The paper concludes with a summary of presented strategies and remaining challenges.

10.2 Current Curricula

This section first reviews typical student preparation for Aerospace Systems careers in precollege (K-12) curricular and real-world exposures. Next, traditional Aerospace Engineering (AERO) and Computer Science and Engineering (CSE) curricula are reviewed in the context of the author's current institution, the University of Michigan. The author recently published a survey of computational course requirements for Aerospace Engineering programs in the USA [1]; results indicate Michigan's current Aerospace curricula are representative as discussed further below.

10.2.1 K-12 Education

Today's youth have cell phones and access to laptop and/or desktop computers. Teens communicate fluently online and very quickly embrace new apps. Higher education courses can therefore assume a student will easily use a Windows environment, downloadable apps, and course web pages. Students will have significant experience with online search engine use and will likely find relevant YouTube and Wikipedia pages faster than faculty. Many students have had "technology" courses in which they learn to use graphics and office software packages. These courses teach navigation through menus, websites, and disk/cloud storage options. However, K-12 does not yet provide consistent coverage of computer science fundamentals, resulting in a significant gap between "usage" of an app and "understanding" of the code needed to realize that app. Because the K-12 curriculum is still struggling to establish a

rigorous and consistent computer science curricular standard, K-12 educators often rely on extracurricular activities such as team-based robotics or gaming competitions to at least attract students willing to devote their spare time to such pursuits.

The current state-of-the-practice in K-12 education has the following results. First, most all K-12 graduates are comfortable with "apps" and can easily communicate and navigate using online resources. Some K-12 students have computer programming experience for activities such as robotics and gaming, but few K-12 students have knowledge of computer science foundations. In contrast, most all K-12 graduates entering a top engineering program will be well versed in math and physics, as evidenced by the fact that most undergraduates arrive at top-tier universities with Advanced Placement (AP) math and/or physics credit.

K-12 students with computer programming exposure certainly capitalize on this exposure in their improved conceptual understanding. However, because programming experiences are primarily extracurricular, no specific foundation can be presumed at a university. What language has a student used, e.g., C++, Java, Python, Basic, JavaScript? Is the student's exposure general or specialized to the particular extracurricular activity, e.g., JavaScript for Minecraft, Arduino C code loops for robots? Because even students with computer programming background in K-12 have such diverse exposures, entry-level college courses cannot assume proficiency in specific procedural or object-oriented code foundations, nor can such courses assume all students have even basic mastery of a common programming language. As a result, freshman programming courses, even within a top college of engineering, have to "start from scratch".

Researchers in K-12 Computer Science (CS) education have long studied the challenges of establishing a standard curriculum ensuring rigorous K-12 student CS preparation. Hubwieser et al. [2] provide a statistical overview of CS education programs internationally, revealing that the struggle to establish a consistent and rigorous K-12 education in CS is worldwide. This study [2] presented statistical results on CS educational terminology, curricular competencies and goals, learning content, assessment strategies, and teacher training requirements. Terminology was varied and somewhat challenging to precisely correlate, a product of the relative youth of CS in comparison to established subjects such as physics. Terminology variance led to large sets of goals and competencies, though aggregation led to "super-categories" such as "Representing, Understanding, Creating, and Testing Algorithms". In the final analysis, the following languages were taught in more than one country surveyed:

- Java,
- C++, C,
- Python,
- AppInventor,
- BASIC, VisualBasic,
- HTML,
- JavaScript, and
- Pascal.

The above list represents an impressively broad suite of languages with different features. Even when a student has had the opportunity to pursue K-12 CS coursework, this breadth in language choices illustrates a major challenge in student preparation inconsistency for entry-level engineering programming courses.

Perhaps more important than programming language is foundational concept coverage. Per Ref. [2] study, the following curricular content was covered in over half of the surveyed countries' curricula: algorithms, applications, computer and communication devices, networks, data structures, database systems, information and digitalization, CS mathematics, modeling, object-oriented concepts, operating systems, problem-solving, and programming. This is an extensive list that sets a high mark for K-12 CS educators; K-12 coverage of this set of competencies could revolutionize our ability to build toward advanced CS concepts in all engineering programs. That said, there remains a substantial gap between the ideals of [2] and the actual K-12 CS preparation observed in students entering US university engineering programs today.

To reflect how significantly the current deficiency in K-12 computer science education impacts a student's educational experience, consider for comparison the current K-12 curricula in traditional mathematics and physics. Any K-12 student aspiring to gain admission to a top-tier university engineering program is advised to take as many math and science courses as they can handle in their K-12 education, with emphasis on basic math skills, algebra, geometry, and calculus. Top students tend to complete all the math available in their high schools, with many adding courses from local community colleges in their senior year. Most high schools now offer some form of calculus, and the majority of students entering a top university earn AP exam credit to place out of at least the first-semester Calculus course. Top students in good schools also pursue AP science credit in physics, chemistry, and biology. This consistent, long-term series of K-12 exposures in math and science provides an undeniable foundation on which college coursework can directly build. For example, even a Calculus I course can assume all entering students understand basic math operations, algebra, geometry, and trigonometry. If a student is not prepared for Calculus I, it is expected the student will "catch up". This is in sharp contrast to the necessary "no student left behind" posture a first-year programming course must take, which ultimately results in a one-term programming course covering only basic conceptual material more akin to Algebra I than to Calculus. As engineering students enter their second year of an undergraduate program, most are still at a low proficiency level in computer programming. As described below, computer science programs incrementally build student knowledge as needed, whereas traditional engineering programs tend to "sweep this deficiency under the carpet" in courses emphasizing math/physics fundamentals. This deficiency can no longer be ignored in preparing the twenty-first-century student for an Aerospace Systems career, regardless of which Fig. 10.1 pillar a student selects.

10.2.2 Traditional Aerospace Engineering (AERO) Curriculum

The traditional Aerospace Engineering curriculum is built on a foundation of K-12 plus first-year engineering math, science, and humanities coursework. Over a century ago, academia recognized the need to split Aeronautics from Mechanical Engineering to enable focus on the atmosphere and flight vehicles in coursework. Researchers envisioned a time when space flight would also be possible, resulting in the creation of Aerospace as well as Aeronautics and Astronautics departments at major universities. Because of its heritage in mechanical engineering, Aerospace fundamentals were organized around topics with analogs in Mechanical Engineering: structures and materials, gas dynamics (a generalization of aerodynamics and propulsion), and vehicle flight dynamics and control. This "pillar structure" for traditional Aerospace Engineering is shown in Fig. 10.2.

Early aircraft designs focused on optimizing aerodynamics and propulsion systems. Structures have evolved from fabric and wood to metal and now to composite materials. Early aircraft hosted mechanical actuation and instrumentation systems. Aircraft and spacecraft have evolved to fly-by-wire vehicle designs, and Unmanned Aircraft Systems (UAS) are fully capitalizing on lightweight, low-cost avionics components. Increasingly-autonomous onboard data-to-decision systems are found on most mass-produced Aerospace vehicles. Still, Aerospace foundations focus on the traditional pillars, though laboratory and design project experiences have evolved to capitalize on modern hardware and software.

The current University of Michigan Aerospace Engineering curriculum has traditional content and is shown in Fig. 10.3. Note that general elective credits are not shown. The University requires the core engineering courses shown in the left column. The next category, Core Aerospace, illustrates the traditional "three-pillar" organization of lecture-based coursework. We currently require a sequence of three laboratory-based courses. Figure 10.3 groups an introductory Aerospace course, a sophomore seminar, and a senior system design course in a single category. Cur-

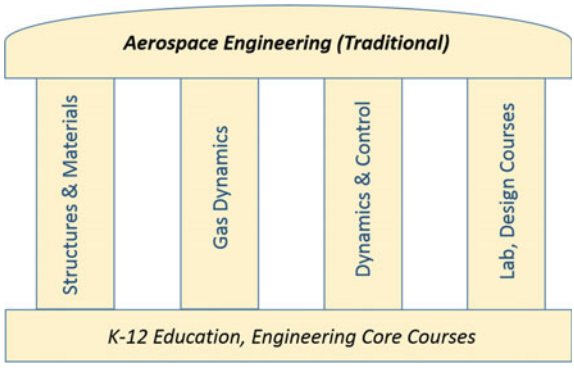

Fig. 10.2 The traditional pillars of Aerospace Engineering

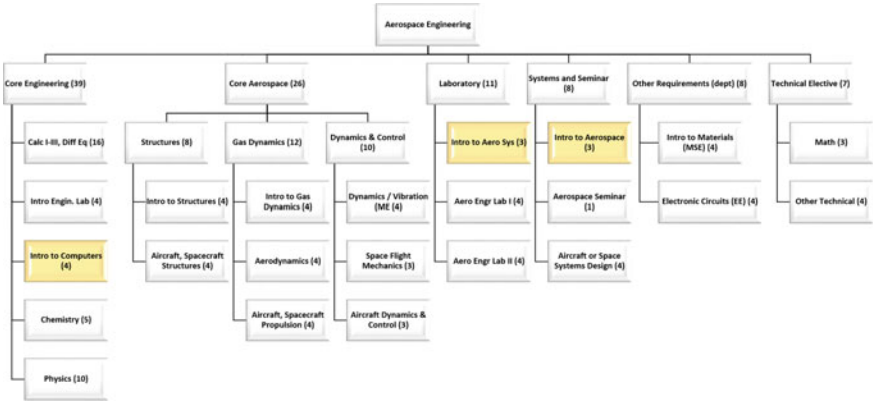

Fig. 10.3 University of Michigan Aerospace Engineering (AERO) curriculum (current)

rently, all Aerospace majors must take an introduction to materials class (taught in Materials Science and Engineering (MSE)) along with an electronic circuits class (taught by Electrical Engineering (EE)), but this is expected to evolve, as discussed later in this paper. The final Aerospace degree requirement, beyond general electives, is a Technical Elective distribution of 7 credits, 3 of which must be acquired in an advanced math course.

As shown in Fig. 10.3, there is very little required CS content in Michigan's current Aerospace Engineering degree program. Required courses containing some CS content are highlighted in the figure. A single freshman programming course introduces engineering students to CS concepts, but as discussed above, this course must cater to students across all K-12 preparation levels. Further, the course needs to cater to all engineering majors, resulting in emphasis on two languages (C++ and Matlab), which further limits coverage depth. This freshman course was never intended to cover CS theory given its main priority to ensure students have at least some programming proficiency for upper-level engineering coursework.

At the sophomore level, two required Aerospace courses require computer programming content. The "Introduction to Aerospace" course, currently taught by the author and two other Aerospace faculty interested in improving the Aerospace CS curricula, has evolved to require Matlab for every assignment and every exam. Early lectures assure students have understanding of the basic language mechanics, important especially for transfer students who have not seen Matlab previously. Assignments then progressively build on numerical methods and plotting capabilities to expose students to the use of a mathematical programming language for aerodynamic, steady flight, orbit, and launch analyses. The "Introduction to Aerospace Systems" laboratory course is structured as a custom hovercraft design, build, test experience in which students learn Computer-Aided Design (CAD), composite manufacturing, and apply basic principles of structural and aerodynamics in their iterative designs. Once students are able to successfully drive their hovercraft via radio-control link,

they are asked to write specific Arduino C control code functionality. These two courses give students some sophomore-level exposure to analysis and embedded code development. These exposures are important, and they reinforce concepts and the languages covered in freshman programming. However, these exposures are not currently followed-through with any further CS content, resulting in limited maturity and confidence for students who choose not to take elective courses involving CS theory or practice.

The Aerospace Engineering graduate curriculum currently follows the same pillar structure organization depicted in Fig. 10.2. While traditional structures, gas dynamics, and control theory courses lay the traditional foundation for students pursuing degrees in this area, specific courses have been added to address computational science and real-time decision-making areas crosscutting with computer science. In gas dynamics, courses in numerical methods and Computational Fluid Dynamics (CFD) expose students to numerical methods and also computational strategies for managing high-performance computing resources needed to solve most important Aerospace flow field problems. In flight dynamics and control, courses in flight software systems and Aerospace information systems have been developed to help students gain further competence in embedded coding for autonomous control, as well as exposing students to CS theoretical underpinning they have not previously seen. The information systems course, in particular, covers concepts from data structures and complexity through finite state automata and discrete search. Our experience is that students find the theoretical underpinnings fairly easy to grasp but find coding projects much more difficult. This finding is not surprising given that students have substantial practice with pencil-and-paper problems in almost every undergraduate Aerospace course, but students have mostly used short Matlab codes to assist with Aerospace problem-solving thus are not well equipped to scale their coding experience to larger projects implemented in a lower level language such as C or C++.

10.2.3 Computer Science and Engineering (CSE) Curriculum

Computer Science and Engineering (CSE) is a relatively new discipline in comparison to Aerospace Engineering. Many universities including the University of Michigan offer two forms of computer science degrees: one within a humanities umbrella (e.g., literature, science, and arts) and another in engineering. New degrees in data and information technology have also emerged in recent decades. Because Aerospace Engineering and CSE have the most natural overlap, CSE is the CS-related degree of reference for this paper.

Figure 10.4 shows required coursework for the University of Michigan CSE program. The left column of core engineering coursework is the same as core coursework for Aerospace except that CSE students are required to take linear algebra in lieu of either Calculus III or Differential Equations. The 24-credit core com-

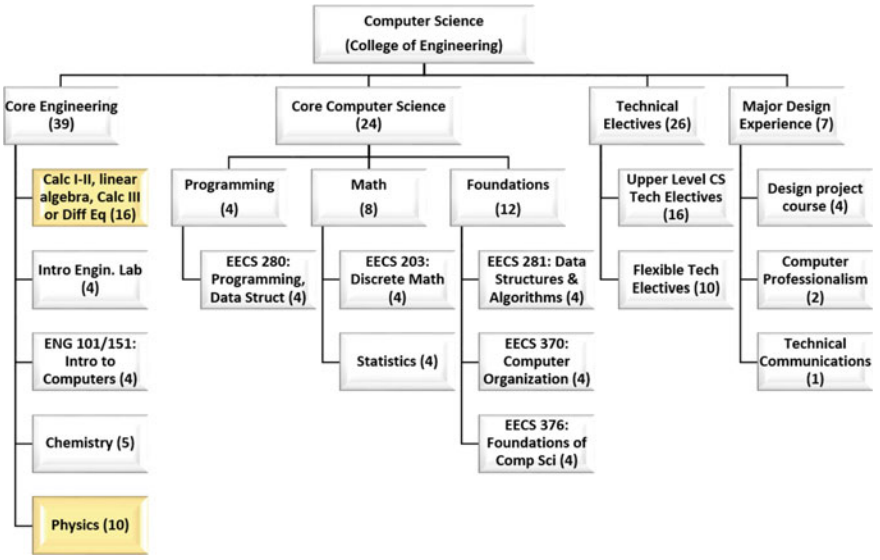

Fig. 10.4 University of Michigan Computer Science and Engineering (CSE) curriculum

puter science required for all students includes fundamental content in discrete math, object-oriented programming and data structures, algorithms, computer organization (introduction to computer architectures), and theoretical foundations. Students learn complexity theory, foundational data structures, and algorithms, and are introduced to grammars, languages, finite state automata, and Turing Machines. CSE students have substantial flexibility in selection of junior and senior level technical electives, and are required to complete a major design experience that typically includes a team-based software development effort along with a formal project reporting (technical communication) requirement.

Figure 10.5 illustrates CSE course prerequisites as a dependency graph.[1] Required sophomore and junior courses are highlighted. As shown, Discrete Math (EECS 203) and the sophomore-level programming course (EECS 280) precede the three remaining required courses (EECS 281, EECS 370, and EECS 376).[2] As shown, there are numerous technical electives available to CSE undergraduates, providing undergraduates substantial specialization opportunities. CSE has experienced remarkable growth in the past decade, so even elective classes tend to have most every classroom seat filled.

Typically, students take EECS 281 (Data Structures and Algorithms) prior to taking either junior-level course. EECS 280 and 281 in particular have lengthy coding project requirements. From the author's experience with undergraduate students,

[1] This chart is reproduced from https://www.eecs.umich.edu/eecs/undergraduate/computer-science/17_18_cs_eng.pdf.

[2] EECS refers to Electrical Engineering and Computer Science.

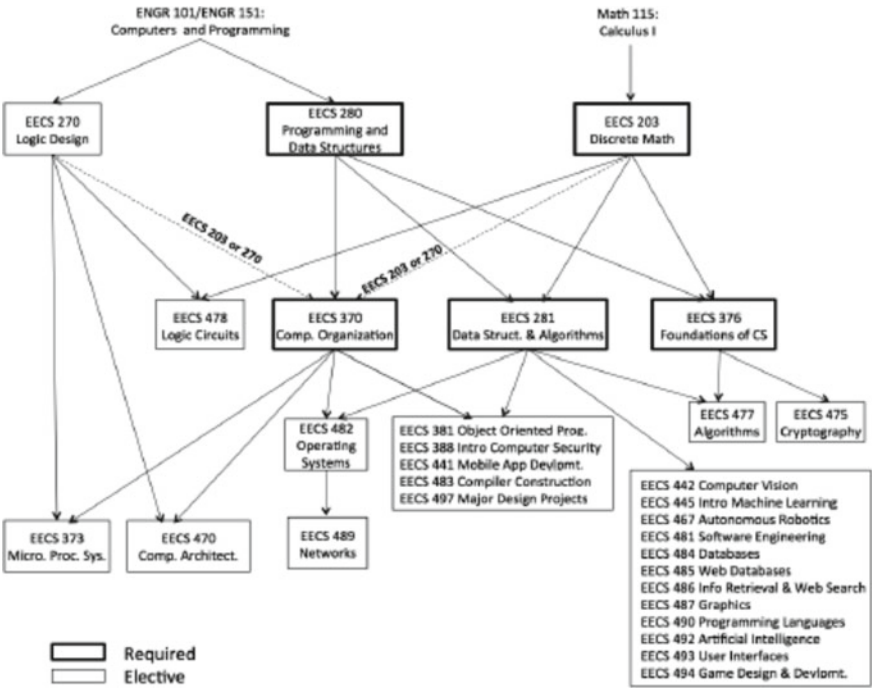

Fig. 10.5 University of Michigan Computer Science and Engineering (CSE) prerequisite map with lists of CSE technical electives

completion of EECS 281 is the "minimum bar" for a student to have reliably achieved a strong foundation in data structures and programming. This implies that non-CSE students seeking competence in CSE will need to take a minimum of 12 credit hours of CSE coursework, EECS 203, EECS 280, and EECS 281, to gain CSE competence.

The Aerospace Engineering department currently recommends students interested in CSE pursue a CSE minor to supplement their Aerospace major. Students pursuing this minor must complete EECS 203, EECS 280, EECS 281, and one of the listed upper-level technical electives. Common choices by Aerospace students include computer vision, artificial intelligence, and autonomous robotics. The CSE minor is a good option for an undergraduate student who either can afford tuition for an extra semester or who arrives with substantial Advanced Placement (AP) credit. However, a CSE minor is not practical for most students following a standard curricular track, suggesting the need for further Aerospace curricular revision.[3]

[3]The University of Michigan Aerospace Engineering Department does not currently offer a minor degree option; if available a student selecting a CSE major and Aerospace minor would face similar course credit challenges.

10.3 Cross-Disciplinary Programs

Two multidisciplinary fields have emerged that have made progress in integrating Aerospace (AERO) and CSE foundations to some extent: Cyber-Physical Systems (CPS) and robotics. Background in each is provided below, although the author acknowledges that in practice neither have targeted Aerospace-specific systems as a focus of attention. Certainly, CPS and robotics program graduates have competence in physics, math, and CSE. However, the Aerospace industry has had trouble attracting the most talented CPS and robotics students due to the extremely favorable market for current CPS and robotics graduates. The high demand for CSE, CPS, and robotics graduates is yet another motivation for augmenting traditional Aerospace degree programs.

10.3.1 Cyber-Physical Systems (CPS)

The CPS focuses on modeling and control of sensing, mobility, information, and networking systems operating in and making decisions about a complex environment [3–5]. 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 researcher studying CPS. CPS research tends to be customized to the domain of interest, from an autonomous vehicle or cooperative vehicle team to collaborative human–robot systems. CPS decisions range from coordination and fault-tolerant control of physical actuators to real-time data management and secure networking. CPS are inherently multidisciplinary, challenging educators and practitioners to build knowledge of fundamental and application concepts that cut across, at a minimum, the fields of dynamics and control (control theory) and computer science (real-time computing).

CPS graduate degree programs have been proposed with some success. In [6], an embedded systems focus is achieved with curricular content in control, communication, distributed systems, machine learning, sensors, and security. Reference [7] emphasizes balance in theoretical and applied CPS coursework and hands-on experience to support "ready to engineer" graduates. In this work, the goal is to graduate CPS engineers with knowledge of physics, software, and systems. Specific education and experience in collaboration across multidisciplinary teams is encouraged. Impact of CPS to society should be recognized in the context of economics, human–machine interfaces, and social/legal contexts. Textbooks in CPS are beginning to be published [8]. Authors have attempted to undertake the challenge of simultaneously providing depth in CS, particularly real-time embedded systems, along with depth in mathematics and control theory for sensor data fusion and robust control.

CPS hold potential to bridge the specific Aerospace—CSE gaps related to computer-based data management and control for complex physical systems. However, most CPS research does not extend consideration to the other aspects of the

Aerospace System, e.g., aerostructural and propulsive systems as well as mission-specific challenges, particularly for space applications.

10.3.2 Robotics

Robotics is a multidisciplinary area of study that integrates principles of mechanical engineering, electrical engineering, and computer science to design, build, and deploy physical systems that assist humans or independently complete physical tasks. Coursework may be characterized in three contexts: sensing, reasoning, and acting.[4] Although plagued by comparisons to science fiction, robotic systems first experienced widespread use decades ago in factory assembly environments. Robotics has experienced rapid growth in the past decade due to affordable microelectronics, capable processors, and high-density energy storage systems. It is now possible to rapidly assemble and test gadgets that drive, fly, swim, and grasp at reasonable monetary and time investment. Robotics programs therefore can introduce students to foundational mathematics and real robotic systems through hands-on manipulator, driving, and flying systems equipped with onboard sensors, including cameras and lidar, augmented by off-board motion capture systems.

The goal of "first courses" in "mathematics for roboticists" and "robotic systems" is to prepare highly capable graduate students with diverse backgrounds for the spectrum of follow-on robotics courses they select. Today, popular robotics subjects include machine learning, machine vision, artificial intelligence and planning, linear and nonlinear control systems, signal processing, mechatronics, and navigation and mapping. New courses in nontraditional areas such as "robot ethics" are also being introduced as issues in policy, law, and culture. While robotics programs currently tend to place primary focus on manipulation, self-driving cars, and human–robot interaction, UAS or "drones" provide rich experiences for students with interest in flight, and robotics principles such as task and path planning and navigation through obstacle fields can also be directly applied to flight vehicles and planetary exploration systems. A number of high-quality robotics textbooks have been published; early texts focused on robotic manipulator kinematics, dynamics, and control. Specialized texts, online courseware, multimedia instructional materials, and open-access databases have been developed for use in robotics education and practice. Of particular relevance are open-source community-developed toolchains such as ROS, the Robot Operating System (http://www.ros.org), and cloud databases such as Open Street Map (OSM) (http://www.openstreetmap.org).

[4]These areas are used to loosely organize the robotics graduate curriculum at the University of Michigan.

10.4 Evolving the Aerospace Engineering Curriculum

This section explores opportunities to evolve the Aerospace Engineering curricula given topics in CSE, CPS, and Robotics that could improve students' expertise in twenty-first-century Aerospace Systems. Certainly, no one, including the author, would like to see key topics in the current curriculum sacrificed, but future Aerospace graduate could benefit from more than the tenuous required CS content included now.

At the core of this challenge are the following key questions:

- What aspects of computer science theory and practice are essential for the Aerospace Engineering student?
- Given credit limits, what do we remove from the required Aerospace Engineering curriculum to "make space" for new content?

The first question is beginning to be addressed through analysis of critical foundations for both Aerospace research and projects in industry. The second question has been more problematic to address, as faculty need to acknowledge that some existing required course content must become elective. Another barrier to evolution is accreditation, with organizations such as ABET (http://www.abet.org) insisting that Aerospace meet the standards required decades ago. Standards must evolve along with curricula to assure relevance in twenty-first-century Aerospace curricula.

This is not the first paper to address extension of the "traditional Aerospace pillars". In 2004, Long [9] proposed a five-pillar structure consisting of (1) fluid dynamics and thermophysics; (2) propulsion and power; (3) structural mechanics and materials; (4) guidance, control, and dynamics; and (5) computing, information, and communication. This publication appears in the inaugural issue of the AIAA Journal of Aerospace Computing, Information, and Communication, now the Journal of Aerospace Information Systems (JAIS). The first four pillars map to the three-pillar structure presented in Fig. 10.2 with fluid dynamics and propulsion both included within gas dynamics. The final or fifth pillar represented what Long considered the essential next step toward a more holistic Aerospace Engineering–CS education. In later publications, Long described a successful Aerospace software engineering course introduced at Penn State University [10], with follow-up publications such as Ref. [11]. The author feels Long's pain, in that she has spent her career pleading for Aerospace curricular reform only to face the curriculum shown in Fig. 10.2 despite over a decade of attempted reform.

Finally, over the past year, steps toward curricular reform have been seriously taken. Though small steps, there is hope at the University of Michigan, a traditional department, which means there is also hope for evolution in other traditional Aerospace programs.[5] Ultimately, the pressure to evolve has not come from tradi-

[5]The Massachusetts Institute of Technology (MIT) has embraced information systems long-term, with some growing pains as traditional faculty felt fundamentals were being sacrificed. Stanford University recently introduced an undergraduate Aerospace degree, which offered them a clean slate in which they were able to include CS content. Other universities have not yet responded; however, it is likely they are feeling similar pressures to the University of Michigan's Aerospace Department.

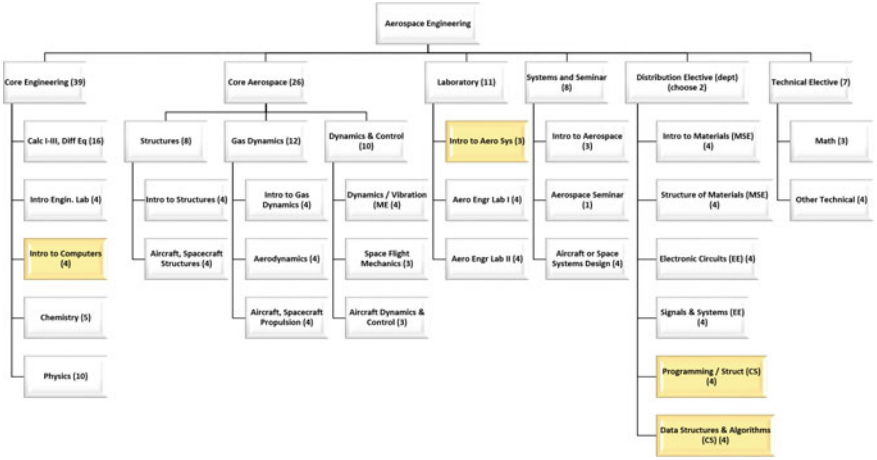

Fig. 10.6 Proposed distribution elective for the University of Michigan's Aerospace Engineering program

tional Aerospace faculty, nor has it come from the Aerospace industry. It instead has come from students, who have increasingly sought CS minors and who have begun to, in some cases, abandon Aerospace as a major due to the challenge of fitting both AERO and CS into a 4-year program of study. A committee of six faculties covering Aerospace autonomy, validation and verification, space systems, and computational fluid dynamics has engaged in proposing specific computing curriculum reforms. To date, there has been little appetite for adding Aerospace courses that fully mirror CS courses, but the faculty has approved a curriculum change that will increase student flexibility, and a new course is being developed to address what are viewed as key gaps in Aerospace student CS background.

Figure 10.6 illustrates a welcome curricular change recently approved by the faculty. Specifically, the two required extra-departmental courses, one in materials and one in electrical circuits, have morphed into a "Distribution Elective" to increase student flexibility. In the revised curriculum, students can select any two of the six listed courses, two from Materials Science and Engineering (MSE), two from Electrical Engineering (EE), and two from CSE. This change will make it easier for Aerospace students to elect a CSE minor in that at least one CSE minor courses, EECS 280 or EECS 281, can also count toward the Aerospace degree. Note that students opting for EECS 281 will have to complete prerequisite EECS 203, but this one additional course will not prevent most students from selecting the CSE distribution elective given interest.

With the modified curriculum, Aerospace students will have the opportunity to deepen their exposures in one or more out-of-department course sequences. However, students who do not elect the EECS 280-281 sequence will graduate with no better CS background than they obtain today. To address this problem, a group of Aerospace faculty are developing a new 4-credit course with a computer laboratory component

that assures one-on-one oversight. The course will be comprised of three CS-centric modules with acknowledged connection to Aerospace: CS fundamentals including data structures, iterative and recursive algorithms, and computational complexity, Computational Science including key numerical methods and efficient memory and computational approaches, and embedded real-time data management and control with a hardware-based final project (quadcopter operating indoors and outdoors in a netted facility). As a junior course, students will continue to build on their Matlab analysis experiences and C/C++ (Arduino) experiences from sophomore year. The course will offer a combination of new materials and repeat exposures to key CSE concepts students might not remember long-term without this new required exposure. We envision a team of two faculties, one with expertise in embedded systems and one with expertise in computational science, to teach the course until clear notes and baseline projects are established. One in place, students will have required course-work exposures to CSE topics at freshman, sophomore, and junior levels. Senior electives in numerical methods and flight software systems will offer students the option of continuing their CSE studies in Aerospace-centric analysis and embedded control.

10.5 Conclusion and Discussion

This chapter has summarized state-of-the-practice in K-12, Aerospace Engineering, and CSE education. Emerging CPS and robotics disciplines were reviewed to offer ideas for extending the traditional Aerospace curriculum to better incorporate the breadth of Aerospace Systems topics, particular computer CSE. CSE augmentations to the University of Michigan's Aerospace program are proposed and discussed.

Exposure of Aerospace students to CSE concepts now appears within reach; however, there are two remaining pillars underlying the twenty-first-century "Aerospace System": electrical engineering and human factors/operations. While consideration of fundamentals for these areas is beyond the scope of this paper, it will be important for Aerospace and CSE programs to continue evolving. Otherwise the gap between "operators" and "engineers" will continue to widen, and Aerospace Engineers might eventually be as unaware of how analog and digital circuitry work as they are of computer science foundations today.

References

1. E.M. Atkins, Education in the crosscutting sciences of aerospace and computing, J. Aerosp. Inf. Syst. **11**(10), 726–737 (2014)
2. P. Hubwieser, M.N. Giannakos, M. Berges, T. Brinda, I. Diethelm, J. Magenheim, Y. Pal, J. Jackova, E. Jasute, A global snapshot of computer science education in K-12 schools, in *Proceedings of the 2015 ITiCSE on Working Group Reports*, ACM, 2015, pp. 65–83

3. R.R. Rajkumar, I. Lee, L. Sha, J. Stankovic, Cyber-physical systems: the next computing revolution, in *Proceedings of the 47th Design Automation Conference*, ACM, 2010, pp. 731–736
4. E.M. Atkins, J.M. Bradley, Aerospace cyber-physical systems education, in *Infotech@ Aerospace (I@ A) Conference*. (American Institute of Aeronautics and Astronautics, USA, 2013), p. 4809
5. E.M. Atkins, Education in the crosscutting sciences of aerospace and computing. J. Aerosp. Inf. Syst. (2014)
6. F. Kurdahi, M.A.A. Faruque, D. Gajski, A. Eltawil, A case study to develop a graduate-level degree program in embedded and cyber-physical systems, ACM SIGBED Rev. **14**(1), 16–21 (2017)
7. M. Törngren, M.E. Grimheden, J. Gustafsson, W. Birk, Strategies and considerations in shaping cyber-physical systems education, ACM SIGBED Rev. **14**(1), 53–60 (2017)
8. E.A. Lee, S.A. Seshia, *Introduction to Embedded Systems: A Cyberphysical Systems Approach* (MIT Press, USA, 2016)
9. L.N. Long, Computing, information, and communication: the fifth pillar of aerospace engineering, J. Aerosp. Comput. Inf. Commun. **1**(1), 1–4 (2004)
10. L.N. Long, O. Janrathitikarn. A new software engineering course for undergraduate and graduate students, in *AIAA Infotech@ Aerospace Conference* 2010, pp. 20–22
11. L.N. Long, On the need for significant reform in university education, especially in aerospace engineering, in *Aerospace Conference* (IEEE, New York, 2015), pp. 1–7