To be presented at: IEEE Aerospace Conference, Big Sky, Montana, Mar. 7-14, 2015

On the Need for Significant Reform in University Education, especially in Aerospace Engineering

Lyle N. Long
The Pennsylvania State University
233 Hammond Building
University Park, PA 16802
814-865-1172
lnl@psu.edu

Abstract—**This paper discusses the need for academia to be more responsive to the needs of students and society. The slow speed of change in academia is causing our educational programs to lose value. Technology has been advancing at an exponential rate for decades, yet academia changes at almost glacial speed. While the example used herein (engineering, and in particular aerospace engineering) is one familiar to the author, the ideas in this paper may well apply to all academic disciplines, including Science and Liberal Arts. The problems are due to rapid changes in technology, inflated bureaucracies at universities, the emphasis on revenue and research, and limits to human learning. There is no question that this is the Information Age, yet academia has not adjusted to this dramatic new world. Students need to understand this and be pro-active to prepare themselves for the future and choose the right courses, majors, and universities.**

TABLE OF CONTENTS

1. INTRODUCTION 1
2. EXPONENTIAL CHANGES IN TECHNOLOGY 2
3. COMPUTER PROGRAMMING.....2
4. AEROSPACE ENGINEERING CAREERS.....3
5. AEROSPACE ENGINEERING CURRICULA.....4
6. CONCLUSIONS.....6
REFERENCES.....7
BIOGRAPHY 7

1. INTRODUCTION

It has been about 7 years since the paper "The Critical Need for Software Engineering Education" [1] was published. In that paper (which should be read before this one) it was explained why we need many more people educated in professional software development. It also discusses why software is crucially important.

In addition, it has been 11 years since the Journal of Aerospace Information Systems [3] was created, which is recognition from the AIAA that computing, information, and communication is a key pillar of aerospace engineering [2]. Unfortunately, there have been few changes in engineering education during this period. In this paper the case will be made more forcefully (and bluntly) that our technological future is not being well served by current universities. Students are being required to learn an enormous amount of very difficult material that they will never use, and conversely, they are not being taught material that will be critical in their future career. While in some fields it is important to teach the detailed history of the subject, in engineering the students need to learn timely, important, and relevant material. Much of what is being taught is taught for historical purposes. This is not fair to our students. They need to learn how to create systems for the information age, not learn how to build systems from 50 years ago.

If aerospace engineering education does not change, it will become as irrelevant as railroad engineering, which would be terrible. Aerospace systems are crucial to the U.S. They represent significant export dollars, they are crucial to U.S. defense, they are crucial for transportation, and they are crucial to exploring the universe.

The issue here is *not* about students getting jobs. There is a shortage of engineers, and so recent graduates can usually get jobs. The real issue is preparing them for the 21st century. For nearly 10 years the author has been surveying job openings at the large aerospace companies (e.g. Boeing, Lockheed-Martin, and Raytheon). The results have been fairly consistent for 10 years. Table 1, presents the number of job openings at Lockheed-Martin in 2011, and is a typical representation of the needs of other companies as well.

	New Graduates	Experienced Engineers
Software Engineering	45	172
Systems Engineering	42	342
Information Technology	25	1041
Mechanical Engineering	6	42
Electrical Engineering	3	48
Aerospace Engineering	1	29

Table 1. Number of job openings at Lockheed-Martin in 2011 [9].

978-1-4799-5380-6/15/$31.00 ©2015 IEEE

Aerospace engineering education programs are clearly not keeping pace with industry if industry does not really need aerospace engineers. They mainly need software engineers, system engineers, and information technology experts.

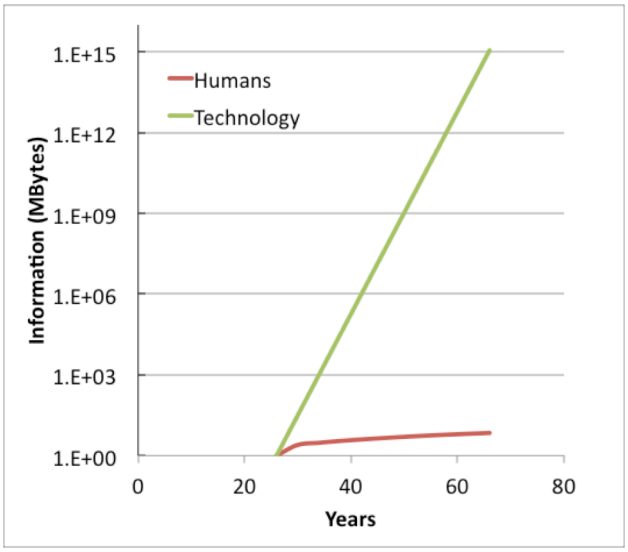


Figure 1. Comparison of linear learning of humans with annual information doubling rate of technology.

2. Exponential Changes in Technology

Technology has been evolving at an exponential rate for hundreds or thousands of years. This has been well described by Kurzweil [4], who discusses the implications of these rapid changes. It is now changing so fast that it is quite difficult for humans to keep up with it. One of the problems is that human learning is linear, while technology changes exponentially. According to Gantz and Reinsel [5] (and others) the amount of online information is doubling every year. In 2020 it is expected that there will be 10^{22} Bytes of information online. The human brain has about 10^{15} synapses and each one can store roughly 1 Byte, so the online data will be roughly 10^{7} times larger than a human brain can hold. Meanwhile, academia has changed little in 500 years.

The implications of this can be illustrated with a very simple "back of the envelope" calculation. Imagine a person reads three books a week and remembers all this information (which is clearly not possible but let's consider it as an upper bound). An average book is about 400,000 Bytes, which gives about 60 Mbytes/year if someone can read and remember the information in three books per week. Figure 1 shows the rate of information growth (doubling every year) compared to the rate of human learning (assuming 3 books per week), where both curves start at unity. This is an over-simplified example, of course, but it helps illustrate the mismatch between human learning and technology change. Over the course of a career, the mismatch between existing knowledge and human understanding grows exponentially. It is important for faculty, students, and employers to appreciate this. We are in the information age, not the industrial age. What professors learned 30 years ago might not be important to current students. And it is extremely difficult for humans to keep up with technology.

The other problem is that making changes in academia is incredibly difficult and slow, especially in large research universities. Any change to the curriculum has to be approved by the departmental faculty, department Head, other departments, the college, the Dean, the faculty senate, the graduate school (for graduate courses), and the Board of Trustees. This process can take several years, which means it will be obsolete before it is approved. Universities need to be able to react more quickly.

3. Computer Programming

Since technology is changing so rapidly, some people do not even appreciate what they do not know. There are still debates in academia about whether we should teach engineers C or Fortran, even though Fortran is a very old and out-of-date language. The Air Force even states it should not be used [6]. But this discussion illustrates how little people know about modern computing and how out of touch they can be.

The IEEE recently evaluated programming languages [7], and there were 23 computer languages that are more important than Fortran. The top four languages were: Java, C, C++, and Python. They evaluated languages in four different ways (i.e. Overall, Job availability, Which are trending, and Which are heavily used for open systems). In all four cases, Java, C, C++, and Python were the most important languages. They also evaluated the languages in four different application areas (Web, Mobile, Enterprise, and Embedded systems). All four of the above languages are heavily used in enterprise computing. Java and Python are heavily used for web applications. Java, C, and C++ are heavily used for mobile devices. And finally, C and C++ (and assembly language) are the key languages for embedded devices. Fortran is no longer important and our students should not be taught it as their first language, yet some academics still debate it's relevance. Incidentally, Fortran is very simple and the students could easily learn it on their own later on if they need it, and Matlab is much more useful than Fortran. The people that ask whether we should be teaching C++ or Fortran don't know what they don't know. The world has moved orders of magnitude away from that question. A good way to assess your programming abilities is to determine where you fit in the programmer competency matrix by Joseph [8]. Many traditional aerospace engineers (including faculty) would not pass the first level.

4. AEROSPACE ENGINEERING CAREERS

I always encourage students to do a job search early in their education. Too often they will get excited about some aspect of engineering, and then when they graduate they find there are no jobs in that area. My 2008 Crosstalk paper [1] discusses which skills and knowledge major aerospace companies (and government labs) are looking for, and it is not the traditional areas of aerospace engineering. They should also search the websites of large and small aerospace companies [9-12]. They should also read the Broad Agency Announcements (BAA) from the major government agencies [13 - 16], which will show them which research areas are important to the DOD, NASA, etc. (and they are *not* traditional aerospace engineering topics).

ABET [17] is another issue. In order for an educational program to be accredited, ABET requires extensive effort by colleges, which is often of limited value. For aerospace engineering programs, ABET [18] states:

> *"Aeronautical engineering programs must prepare graduates to have a knowledge of aerodynamics, aerospace materials, structures, propulsion, flight mechanics, and stability and control."*

These are the "traditional" technology pillars [2] of aerospace engineering. Unfortunately, this is how aerospace engineering was described 50 years ago, when aircraft had no onboard computers, networks, or data. Students should be able to design and build the aerospace systems of the future, not reproduce ancient technology. Today, without detailed knowledge of computers, software, networking, embedded systems, and software and systems engineering; the above curricula would not be of much value. The ABET description mentions nothing about aerospace engineers learning about computers or software. This is ridiculous. All the above topics are very mature compared to computers and software, and few jobs exist in these areas. Fortunately for students, there is a big need for engineers, and employers often train new employees in areas they are not well versed in. So the whole complicated and extremely time consuming ABET process for aerospace programs is based upon erroneous assumptions.

In addition, if someone wanted to work in one of these areas (aerodynamics, aerospace materials, structures, propulsion, flight mechanics, and stability and control), they would have to know a great deal about computers also. Aerodynamics and propulsion work is now performed on massively parallel Linux-based computers using computational fluid dynamics, which is base upon numerical methods and mathematics. Likewise, the behavior of materials and structures are determined using finite element analyses on Linux-based parallel computers. Stability and control requires extensive understanding of mathematics, and usually requires the use of Matlab and Simulink. So today even the original pillars of aerospace engineering are built upon computers and software. Even experimental techniques require one to understand modern instrumentation, data collection, computers, and software.

To put this in perspective, the next generation of long-range bombers will cost more than $500 million each, and 50-70% of this cost will be in computers and software, which is typical of modern aerospace systems. Yet in a typical Aerospace Engineering program, only about 5% of the curriculum is devoted to computers and software. Therefore the students cannot appreciate or even contribute to the entire system. They may not even be able to communicate technically with the majority of the other engineers. If Aerospace Engineering is not modernized, it will become irrelevant. We should not be training engineers how to design and build aircraft from the 1950's.

Software has been called the Achilles' heel of aerospace engineering [19], but yet it is hardly taught in aerospace engineering programs. The Boeing 777 has more than 1000 processors onboard and nearly 20 million lines of code. Most of the problems that occur on new aircraft and spacecraft are software related, yet software engineering is not normally part of aerospace engineering.

Also, one of the most rapidly growing areas in aerospace is drone technology (or UAV's). It is very easy to buy unmanned radio-controlled aircraft with efficient engines and propellers [20]. You can also buy autopilots off the shelf [21]. Therefore, anyone can buy a drone and the aerodynamics, structure, propulsion, and control system is already done. Again, all the "traditional" aerospace disciplines are mature. The difficult aspects of drones are mainly in computers, software, sensors, signal processing, artificial intelligence, networking, navigation, etc.; almost none of which is being taught, especially to aerospace engineering students. When the author first started working on UAV's (about 10 years ago) he was shocked to realize that most of the aerospace engineering students were of little value, since we needed people who understood software, sensors, numerical methods, and computers. We could buy the aircraft, engines, and autopilots.

In addition, some of the key problems in designing UAV's are completely ignored in most aerospace programs: fault tolerance, single-point failure analysis, command and control, radio frequency (RF) links, and compatibility with Next-Gen systems.

Similar stories could probably be presented for most other disciplines (engineering, science, liberal arts, etc.). We need dramatic and significant changes in academic curricula; incremental changes amidst exponentially fast technology change will not work. Our students deserve better. <u>All students (no matter what their major is) would be wise to obtain a Minor related to computing and software.</u>

Often the students understand these issues better than the faculty. One institution (which will go nameless here) started offering an undergraduate minor in computer science. It was so popular that they could not handle the

number of students and cancelled the Minor program. Obviously academia is not run like a traditional business The University should have put more resources into the program to keep it alive.

5. AEROSPACE ENGINEERING CURRICULA

In my experience it is almost impossible to change academic curricula. Too many people have vested interest in keeping the status quo, and there are few financial incentives to do it. There is a way around the impasses however, by creating undergraduate and graduate Minor degrees. These allow students to get the educational components they need, and they get credit for them. They are also much easier to create than it is to modify existing Majors, however they are not usually well funded.

In about 1999 the Penn State Graduate Minor in High Performance Computing was created [22]. This was to address the need for scientists and engineers to learn numerical methods and parallel programming. Since then it has been renamed the Graduate Minor in Computational Science. Information on this program, courses, students, etc. can be found on the webpage [22]. We have graduated 248 graduate students, who are now working around the world, and we currently have 86 more enrolled. These students come from across the university, not just engineering. One student told me that the Minor was more important to her career than her PhD major (chemical engineering). Another Ph.D. student said sample codes from my courses are now part of weather forecasting codes. A Ph.D. student in astrophysics said my course was the most useful one he had had in graduate school.

Building upon the success of the Graduate Minor, in 2006 an undergraduate Minor in Information Science and Technology (IST) for Aerospace Engineers was created. This is described on a webpage also [23]. Undergraduates need to complete 19 credits in computing, networking, information, and software to receive this minor. But it is relatively easy for them to obtain this Minor, since some of the courses (about 9 credits) can be counted towards the major as well. Two new courses (Advanced Computer Programming [24] and Software Engineering [25]) were also created that can be used towards this Minor. Again, the students usually know what is important. These courses are always full, and sometimes there is a waiting list. Each year there are 50-60 students in these courses. Undergraduate students have told me these courses helped them in getting good jobs. Employers are usually very interested in aerospace engineering students who also know about computing and software.

Of course there are many undergraduate and graduate minors offered at most universities. Every student should obtain at least one minor. Too often they obtain minors that are too similar to their major however, which is a mistake. Some excellent minors students should consider are mathematics, statistics, business, computer science, information technology, etc. But every student (in all majors) would be well served by obtaining a minor in computing and/or software. In addition, most computer science graduates are not educated in software engineering. The graduate and undergraduate minors have been effective, but too often students are being asked to learn material in their major that they will never use. In order to spark a debate about this, this paper will discuss the curriculum in more detail, and suggest updates.

2 shows the current courses required at Penn State for a degree in aerospace engineering, which is quite similar to other programs of this kind around the world. The numbers indicate credit hour requirements. A 3 credit course meets three times a week for 50 minutes each. The degree requires 131 credits in total. They need to take 31 credits in the Liberal Arts, which is important (and also a rigid requirement). They need 20 credits of Math, which includes calculus, ODE's, PDE's, linear algebra, complex variables, etc; and 10 credits in physics. They do not have to take any course in Systems Engineering, and they are only required to take 3-6 credits in computing and programming. The traditional (and mature) topics (aircraft or spacecraft design, aerodynamics, propulsion, stability & control, lab, and structures) require 52 credits. The remaining 12 credits are technical electives. This program is far too light in computing and software, and too heavy in the traditional areas. It also gives the students a false impression of what is important in the 21st century.

The courses required for the Aerospace Major combined with the Minor in IST are shown in Figure 3. At a minimum they need 10 additional credits (for a total of 141), but in this approach they take 22 total credits in computing-related courses. And if they schedule their courses carefully they can still graduate in four years (many of the IST courses are offered in the summer and/or online). They still have the same number of credits in the traditional aerospace courses, liberal arts, math, and physics. And they still have 6 credits available for electives, which they should choose carefully (i.e. they should choose more courses in areas such as computing, numerical methods, control theory, robotics, or mathematics).

Figure 4 shows a new recommended curricula for aerospace engineering, just for discussion purposes. It assumes the number of total credits must remain at 131 (this is typically very difficult to change at a university). It is unlikely, unfortunately, that this would ever get approved by an existing aerospace engineering department. In this curricula there are 21 required credits in computing and software, and 3 more credits in math (for numerical methods). The traditional areas of design, structures, and fluids have been reduced by 6, 3, and 3 credits, respectively. The technical electives have been reduced as well. In addition, a course in Systems Engineering would be required. These students would have a tremendous foundation on which to build their careers in the 21st century and would be highly sought after by industry, government labs, an graduate schools. In addition, they could easily complete this in four years.

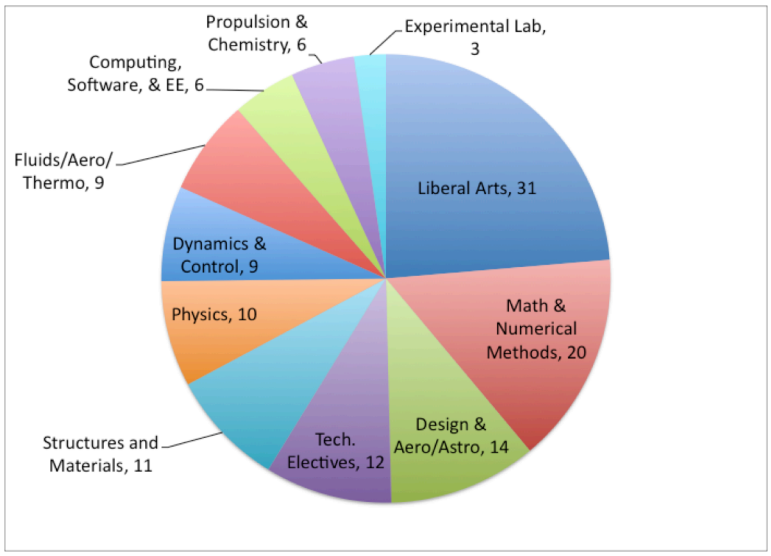


Figure 2. Breakdown of credits required for Aerospace Engineering degree at Penn State University.

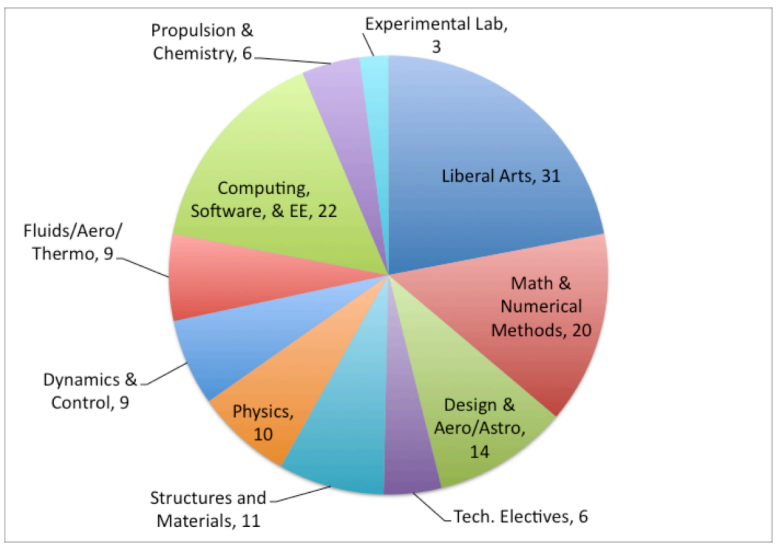


Figure 3. Breakdown of credits required for Aerospace Engineering degree with IST Minor at Penn State University.

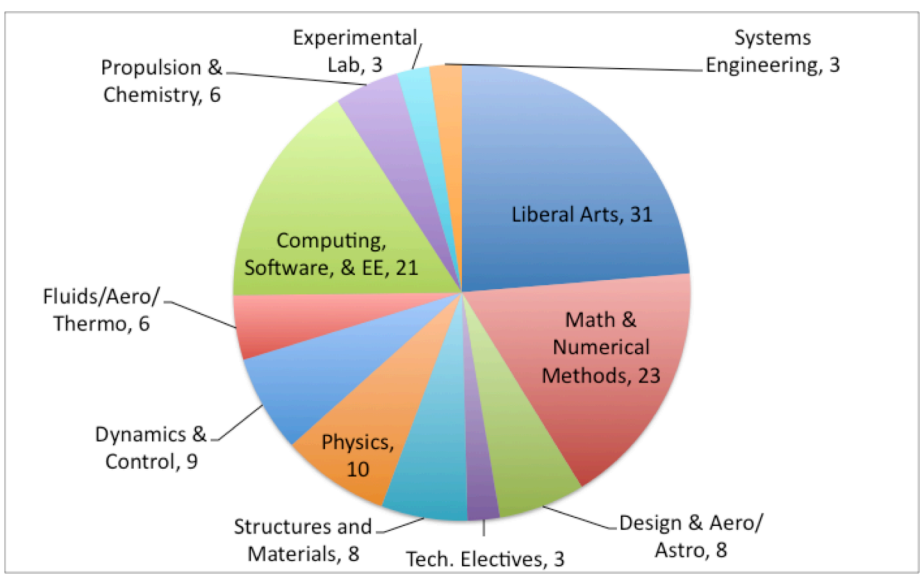


Figure 4. Breakdown of credits required for recommended Aerospace Engineering degree.

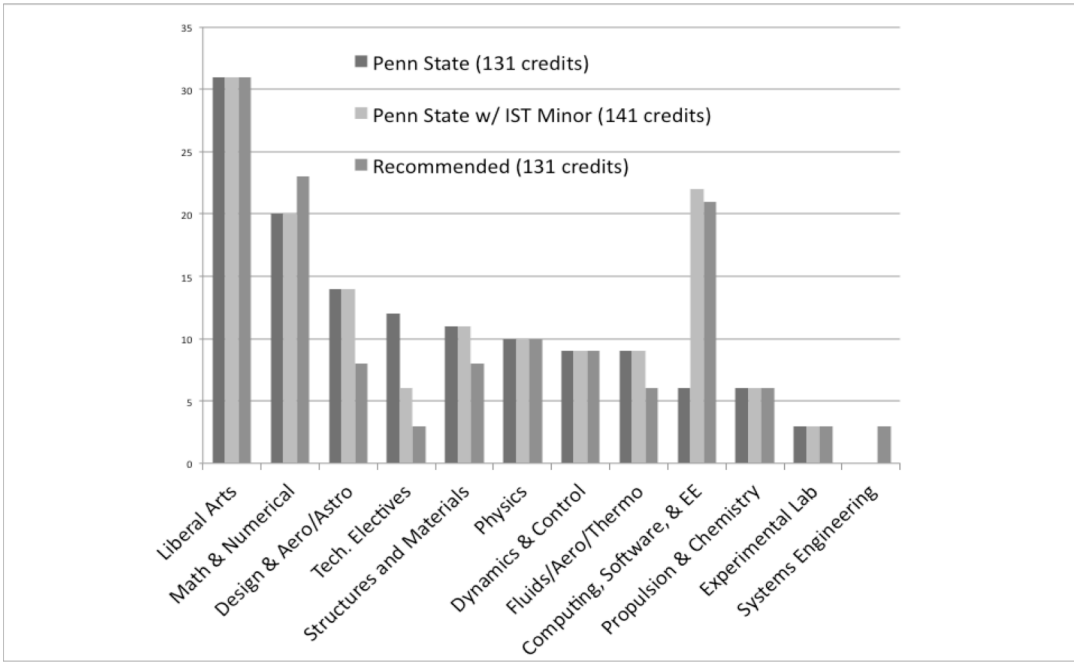


Figure 5. Comparison of the three different curricula (current, current plus IST Minor, and recommended).

Figure 5 summarizes the three different curricula, and makes it easer to see the key differences.

In most of this paper reference has been made to the topic "computers and software." It is important to describe this in more detail. Some of the topics that would be incredibly valuable for students (and their future employers) would be:

Undergraduate:

1. C/C++
2. Web programming (Java and HTML)
3. Python
4. Matlab & Simulink
5. Linux (including real-time operating systems, RTOS)
6. Numerical methods and discrete math
7. Electrical Circuits and Microprocessors
8. Robotics or UAV's
9. Software Engineering
10. Systems Engineering

Graduate:

1. Advanced programming
2. Networks
3. Embedded devices
4. Intelligent Systems /Artificial Intelligence
5. Neural networks
6. Navigation (including GPS)
7. Avionics
8. Electromagnetics
9. Parallel computing
10. Statistics and Big Data

This is a long list of topics (and not complete). The first ten items should be included in any undergraduate program in aerospace engineering. The first six items should be learned by almost all students in any major. If they do not, then the students need to find a way to learn them on their own (unfortunately). Many of the other ones could be incorporated into M.S. or Ph.D. programs (which we have done in our Graduate Minor program [22]). The students would be much better able to tackle these in graduate school if they have a 21st century undergraduate program to build upon. All undergraduate students should obtain some sort of Minor in computing or software. It is not recommended that students Major in software engineering, it is better as a Minor or as a graduate degree.

6. CONCLUSIONS

Computers and software are crucial parts of aerospace systems, and the largest and most important part. Too often traditional aerospace engineers (including faculty) limit their consideration of computing and software as tools, and not part of the system. Faculty are often poorly trained in computing and software, since the technology changes so rapidly. Academic programs (in all disciplines) are very often badly out of date because they are difficult to change, but if the U.S. is to remain preeminent in engineering and science we must modernize our curricula. This paper has mainly addressed undergraduate education, but similar issues exist in graduate programs as well. Technology has been changing at an exponential rate for a thousand years, while academia has changed little in 500 years. Students need to understand this and take charge of their education. If Universities do not modernize their curricula, students must look for appropriate Minors (e.g. [22] and [23]) which will give them 21st century skills.

REFERENCES

[1] Long, Lyle N., " The Critical Need for Software Engineering Education," in CrossTalk: The Journal of Defense Software Engineering, Vol. 21, No. 1, Jan, 2008.

[2] Long, L.N., "Computing, Information, and Communication: The Fifth Pillar of Aerospace Engineering," Editorial, Journal of Aerospace Computing, Information, and Communication, Vol. 1, No. 1, Jan., 2004.

[3] AIAA JAIS Journal, http://arc.aiaa.org/loi/jais

[4] Kurzweil, R., "The Singularity Is Near: When Humans Transcend Biology, Penquin Books, 2006.

[5] Gantz, J. and Reinsel, D., "The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the Far East," Dec., 2012. [http://idcdocserv.com/1414, viewed July 9, 2014.]

[6] Air Force Gentle Introduction to Software Engineering (GISE), www.stsc.hill.af.mil/resources/tech_docs/gise.doc

[7] Stephen Cass, S., Diakopoulos, N., and Romero, J.J., The Top Programming Languages, IEEE Spectrum, July, 2014.

[8] http://sijinjoseph.com/programmer-competency-matrix/

[9] http://search.lockheedmartinjobs.com/

[10] http://jobs-boeing.com/

[11] http://jobs.raytheon.com/search/

[12] avportal.avinc.com/CareersPortal/JobOpenings.aspx

[13] http://www.onr.navy.mil/contracts-grants/funding-opportunities/broad-agency-announcements.aspx

[14] http://www.arl.army.mil/www/default.cfm?page=8

[15] http://www.wpafb.af.mil/library/factsheets/factsheet.asp ? id=8127

[16] http://nspires.nasaprs.com/external/solicitations/summary.do?method=init&solId={7B28F838-C20E-7C9A-E0E1-0F170B331AE9}&path=open

[17] www.abet.org

[18] http://www.abet.org/eac-criteria-2014-2015/

[19] http://chess.eecs.berkeley.edu/hcssas/index.html

[20] Brian R. Geiger, Joseph F. Horn,. Anthony M. DeLullo,. and Lyle N. Long, and Al F. Niessner, "Optimal Path Planning of UAVs Using Direct Collocation with Nonlinear Programming," AIAA Guidance, Navigation, and Control Conference, Keystone, Colorado, Aug., 2006.

[21] http://www.cloudcaptech.com/

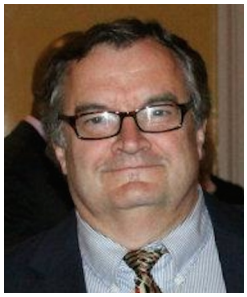
[22] http://www.csci.psu.edu/

[23] http://www.personal.psu.edu/lnl/ist/

[24] http://www.personal.psu.edu/lnl/424pub/

[25] http://www.personal.psu.edu/lnl/440pub/

BIOGRAPHY



Lyle N. Long *received a B.M.E. degree from the University of Minnesota, Minneapolis, MN, the M.S. degree from Stanford University, Stanford, CA, and the D.Sc. degree from George Washington University, Washington, DC. He is a Distinguished Professor of aerospace engineering computational science, neuroscience, and mathematics at The Pennsylvania State University, State College, PA, USA. He is the Founder and Director of the Graduate Minor Program in Computational Science. He was the Founding Editor-in-Chief of the AIAA Journal of Aerospace Information Systems. From 2007 to 2008, he was a Moore Distinguished Scholar with the California Institute of Technology, Pasadena, CA, USA. He was previously with Lockheed Aircraft (Burbank, CA) and Thinking Machines Corporation (Cambridge, MA). He has authored more than 250 journal and conference papers. His current research interests include neural networks, software engineering, cognitive robotics, high performance computing, and unmanned vehicles. Dr. Long received the Penn State Engineering Society Outstanding Research Award in 1996, the 1993 IEEE Computer Society Gordon Bell Prize for achieving highest performance on a parallel computer, and the Lockheed Aeronautical Systems Company Award for excellence in research and development. He is a Fellow of the American Physical Society and the American Institute of Aeronautics and Astronautics.*