A New Software Engineering Undergraduate Program Supporting the Internet of Things (IoT) and Cyber-Physical Systems (CPS)

Prof. Linda M Laird, Stevens Institute of Technology (School of Systems & Enterprises)

I am currently an industry professor in software engineering at Stevens Institute of Technology. Prior to that, I ran large development projects at Bell Labs/Lucent.

Dr. Nicholas S Bowen, Stevens Institute of Technology

Dr. Nicholas Bowen is an Industry Professor in the School of Systems and Enterprises. His primary focus is developing new graduate programs that combine Systems Engineering & Software Engineering with Cyber-Physical Systems.

He recently retired from IBM after a 31-year career. He held a diverse set of leadership positions across product development (both hardware and software), supply chain and manufacturing, sales operations, research, corporate strategy, leading large teams, and talent development. Nick has led and contributed to many critical projects including saving the mainframe business, taking AIX/Power to the #1 UNIX position, establishing Linux servers in the enterprise market, and was on the team that built the first Bladed architecture for the general purpose x86 market.

Nick received a Ph.D. in Electrical and Computer Engineering from University of Massachusetts at Amherst, an M.S. in Computer Engineering from Syracuse University, and a B.S. in Computer Science from the University of Vermont.

Nick has been on the advisory boards of many engineering schools including Florida International University, North Carolina State University, University of Puerto Rico (Mayaguez) and the URI Research Foundation. He was a founding member of the IEEE Computer Society Industrial Advisory Board.

Nick is an avid sailor having logged over 5,000 miles in the open ocean and is often found competing in sailboat races.
A New Software Engineering Undergraduate Program, supporting the Internet of Things (IOT) and Cyber-Physical Systems (CPS)

Linda Laird, Nicholas Bowen
Stevens Institute of Technology
Hoboken 07030, USA
linda.laird@stevens.edu
nicholas.bowen@stevens.edu

Abstract— In the fall of 2015, Stevens Institute of Technology welcomed the first freshmen into a newly launched Software Engineering Undergraduate Program based largely on the most recent ACM and IEEE-CS guidelines for undergraduate software engineering programs [1]. This is the first such program in the US that also has an ABET accredited general engineering curriculum. Students will receive a B.E. in Software Engineering Degree (SWE), and be prepared to sit for the Fundamentals of Engineering (FE) examination [2].

More recently, Stevens’ faculty have begun to study the technical competencies specifically required for the engineering of Cyber-Physical Systems (CPS) and the Internet of Things (IOT). Our preliminary result is this curriculum can provide most of the additional technical competencies required to be an effective software engineer and systems architect for complex CPS and IOT systems and well as the vital interdisciplinary experiences.

This paper describes the program and curriculum, posits additional competencies required for effective CPS/IOT software engineering, and concludes that the new SWE program is a good match for a CPS/IOT software engineering program.

Index Terms—software engineering education, software engineering curriculum, Cyber-Physical Systems, Internet of Things, undergraduate engineering education

I. Introduction

Software Engineering is critical to the 21st century. William Scherlis wrote, “…software has become the building material of choice for nearly all complex systems” [3]. As systems become increasingly more complex, rendering ad-hoc development inadequate, Software Engineers are becoming the builders of the 21st Century. This statement is not meant to imply that other engineering professions are unimportant, only that software is becoming more dominant. Consider a few domains: automobiles (driverless cars), medical (remote medicine), and home appliances (Roomba, Nest). In all of them, a large portion of recent significant “advances” and competitive differentiators are implemented in software. And in many of them, the systems are either Cyber-Physical or Internet of Things systems (CPS/IOT).

Meanwhile, on any given day, there are over 150,000 job openings for software engineers in the United States [4]. These openings represent a huge variety of jobs, ranging from Python programmers or network administrators to overall systems architects. Frequently jobs may be called software engineering, while in reality, they require little more than software programming or network administration skills. On the other hand many of these jobs, especially the more
difficult and complex ones, are for software engineering of very complex systems, such as guidance control systems, supply chain management, and sophisticated web applications. They require architecture, design, testing, and project management skills not typically taught in more programming-focused programs or the recently popular boot camps.

Stevens has offered a Master of Science (MS) in Software Engineering since 2001. It is based on the IEEE/ACM SWE curriculum guidelines for graduate programs in software engineering [5] and is similar to many other schools. The program covers the full development life cycle, is quantitative and empirical, and emphasizes the engineering and development of trusted systems. In this context, the “engineering” in software engineering means using good engineering judgment to select the appropriate architecture and designs, tools and techniques, and applying them to build dependable software.

Stevens is now offering a Bachelor of Engineering (B.E.) in Software Engineering. This new undergraduate program goes beyond the M.S. program in adding the foundation of traditional engineering to the software engineer [6]. It strengthens the Engineer in Software Engineer.¹ This is a unique program in that the graduates will be qualified to sit for the F.E. exam and eventually the P.E. exam. In addition, it is ABET accredited², it meets the recent ACM/IEEE SWE undergraduate guidelines, and even includes systems engineering course work. The graduates should be able to effectively build a variety of challenging systems, utilizing their broad engineering and their deep software engineering skills.

The remainder of this paper is organized as follows:

- **Section II: The Stevens Undergraduate SWE B.E. Program.**
- **Section III: Cyber-Physical Systems and the Internet of Things.**
- **Section IV: CPS/IOT Reference Architectures and Models.**
- **Section V: Competencies for the CPS Software Engineer.**
- **Section VI: Comparison of CPS SWE to Stevens Undergraduate B.E. in Software Engineering.**
- **Section VII: Additional Work on CPS Software Engineering Curriculum.**
- **Section VIII: Summary.**
- **Section IX: Acknowledgements.**
- **Section X: References.**

## II. Stevens’ Undergraduate Software Engineering Bachelor of Engineering Program

The SWE curriculum is a standard Stevens Engineering Undergraduate Curriculum, consisting of 141 total credit hours. The SWE program-specific content consists of 9 required courses. The results to date are very encouraging. The fall 2015 enrollment is 18 freshman, 5 sophomores, and one junior. We expect a small number of graduates in 2018, with the number growing to 20 or 25 per year by 2022. The program goal is to remain a small, high-touch program.

¹ This statement does not imply that all software engineers should have an engineering background. There are many pathways to software engineering, and many different careers with different requirements.

² It is currently accredited under the school’s ABET accreditation. Once there are sufficient graduates, it will be accredited in software engineering.
A. Motivation for Program

Stevens’ Mission is to “inspire, nurture and educate leaders in tomorrow’s technology-centric environment while contributing to the solution of the most challenging problems of our time.” Modern society depends upon systems of increasing complexity to sustain our quality of life, and the engineered systems being conceived and developed today have an increasing and significant percentage of their functionality allocated to software. This requires our future engineers to be strongly rooted in the fundamentals of engineering and science, while also being equipped with a strong capability to develop and integrate software as a central feature in these systems. The software aspect of engineered systems today is not just another component of the system. It also provides the overarching integration framework, allowing systems to be both rich in functionality and capabilities, while being adaptive to context and control. This educational program also includes course work in systems thinking with electives that permit students to explore a domain of interest – domains such as healthcare, embedded systems, financial systems, naval engineering, or control systems.

B. Curriculum:

The Stevens Undergraduate Engineering Curriculum is a rigorous, hands-on curriculum. There are 9 required SWE courses (27 credit hours), 2 courses for Senior Design (6 credit hours), and 2 courses for Domain Electives (6 credit hours). Of the 9 required courses, 5 are significantly new courses and 4 are adapted from the Software Engineering Master’s Program.

Two distinctive aspects of the Stevens engineering curriculum are the traditional breadth of engineering education (see Figure 1) and the integrative, eight-course Design Spine. The Design Spine is a fundamental component of the engineering curriculum that is required for all B.E. students, regardless of discipline. It consists of eight core design courses taken throughout all eight undergraduate semesters of study including a two-semester capstone senior design project, which introduces students to the underlying principles of engineering design through hands-on and project-based learning [7]. For the software engineering students, the senior design project will be a multi-disciplinary project focused on the students’ domain of interest. The Design Spine causes broad exposure to the many fields of engineering with hands-on problem solving experience, which in itself is critical in preparing the students to succeed in the workplace. Students learn and practice interdisciplinary skills through their projects.
The 9 required SWE courses are shown in the table below.

<table>
<thead>
<tr>
<th>Course Number</th>
<th>Course Name</th>
<th>Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSW 215 (New)</td>
<td>Individual Software Engineering</td>
<td>3</td>
</tr>
<tr>
<td>SSW 315 (New)</td>
<td>Object-oriented software development</td>
<td>3</td>
</tr>
<tr>
<td>SSW 322 (New)</td>
<td>Software Design and Evolution</td>
<td>3</td>
</tr>
<tr>
<td>SSW 345 (New)</td>
<td>Model-Based Software Engineering</td>
<td>3</td>
</tr>
<tr>
<td>SSW 533 (Adapted)</td>
<td>Software Estimation and Measurement</td>
<td>3</td>
</tr>
<tr>
<td>SSW 555 (Adapted)</td>
<td>Agile Methods for Software Development</td>
<td>3</td>
</tr>
<tr>
<td>SSW 564 (Adapted)</td>
<td>Software Requirements Engineering</td>
<td>3</td>
</tr>
<tr>
<td>SSW 567 (Adapted)</td>
<td>Software Testing and Quality Assurance</td>
<td>3</td>
</tr>
<tr>
<td>SYS 481 (New)</td>
<td>Systems Engineering and Architecture</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 1

These required courses cover all the phases of software development and maintenance. The first three courses teach fundamental skills of computing, while later courses cover specific lifecycle activities, such as requirements, architecture and testing. All the courses emphasize teamwork and the role of software in larger systems.

The curriculum follows the standard Stevens Engineering curriculum as shown below:

<table>
<thead>
<tr>
<th>TERM I</th>
<th>TERM II</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH115 – General Chemistry I</td>
<td>MA123 – Series, Vectors, Functions, and Surfaces</td>
</tr>
<tr>
<td>CH117 – General Chemistry Lab I</td>
<td>MA124 – Calculus of Two Variables</td>
</tr>
<tr>
<td>MA121 – Calculus IA: Differential Calculus</td>
<td>PEP111 – Mechanics</td>
</tr>
<tr>
<td>MA122 – Calculus IB: Integral Calculus</td>
<td>MGT103 – Introduction to Entrepreneurial Thinking</td>
</tr>
<tr>
<td>E101 – Engineering Experience I</td>
<td>E122 – Engineering Design I</td>
</tr>
<tr>
<td>E120 – Engineering Graphics</td>
<td>CAL 105 or 103</td>
</tr>
<tr>
<td>E121 – Engineering Design I</td>
<td>Science Elective</td>
</tr>
<tr>
<td>E115 – Introduction to Programming</td>
<td></td>
</tr>
<tr>
<td>CAL 103 or 105</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TERM III</th>
<th>TERM IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>MA221 – Differential Equations</td>
<td>MA134 – Discrete Math</td>
</tr>
<tr>
<td>PEP112 – Electricity and Magnetism</td>
<td>E232 – Engineering Design IV</td>
</tr>
<tr>
<td>E126 – Mechanics of Solids</td>
<td>E234 – Thermodynamics</td>
</tr>
<tr>
<td>E245 – Circuits and Systems</td>
<td>SSW215 – Individual Software Engineering</td>
</tr>
<tr>
<td>E231 – Engineering Design III</td>
<td>Science Elective</td>
</tr>
<tr>
<td>Humanities</td>
<td>Humanities</td>
</tr>
</tbody>
</table>
II. Program Outcomes and Assessment:

Graduates of this program will:

- Employ sound principles and practices to design and implement software for complex engineered systems.
- Assume a variety of roles on multidisciplinary engineering teams.
- Communicate effectively with stakeholders in oral, written, and newly developing modes and media.
- Demonstrate professionalism, including continued learning and professional activities.
- Contribute to society by behaving ethically and responsibly.

The achievement of these outcomes and the interpretation of results will be assessed using the standard Stevens processes for engineering programs, including the student grades and their accomplishments in Senior Design. Yearly program reviews, for at least the first five years, will be held to review the achievement of the outcomes and to determine improvements for the following year. These reviews will be based on student surveys and faculty assessments of learning outcomes.

A. Differentiating SWE from Computer Science and Computer Engineering

Differentiating this computing program from the other computing programs at Stevens, both internally and for students and parents, was challenging. After many false starts, including trying to use a 2005 IEEE/ACM model [8] made for this purpose, we built the model below, primarily for internal use.

Software Engineering (SwE), Computer Science (CS), and Computer Engineering (CE) are all intersecting but distinct disciplines, distinct programs, with distinct curriculums. This model
compares and contrasts these programs based upon the coverage of six knowledge areas; 0 indicates no coverage and 5 indicates maximum coverage. It demonstrates that there are both intersections and unique aspects of each program.

III. Cyber-Physical Systems (CPS) and the Internet of Things (IOT)

CPS and IOT are defined slightly differently by different sources, as well as the relationship between them. CPS and IOT are obviously related; some call them equivalent, some believe IOT to be a subset of CPS (in which the network or communications portion is the Internet), and some advocate that IOT is the underlying, enabling technology upon which CPSs are implemented. The last view is the one we have adopted, although including non-internet networking mechanisms.

This uncertainty of definitions is expected: NIST has a CPS working sub-group whose charter is to “develop a common CPS vocabulary, including a working definition of CPS and descriptions of common features, capabilities, and characteristics, to inform the development of CPS reference architecture(s) and to promote communications across CPS stakeholders and domains [9].”

NSF defines Cyber-Physical Systems (CPS) as “engineered systems that are built from, and depend upon, the seamless integration of computational algorithms and physical components. Advances in CPS will enable capability, adaptability, scalability, resiliency, safety, security, and usability that will far exceed the simple embedded systems of today [10]” (emphasis added).

Our goal is not to mediate the debate over CPS vs. IOT, but rather create a new educational experience that builds important skills applicable in both domains while the debate continues. The internet-of-things finds many of its origins in an IBM Global Technology Outlook done in 2007 that identified “Real World-Aware” as a key future theme [11]. This work observed that internet-connected devices were collecting vast quantities of data that immense potential value but was often being thrown away. With better filtering at the edge of the network and better analytics the data could create powerful closed loops systems for health care, public safety and many other applications.

There are many examples that fit the CPS definition: driver-assisted cars, ATM machines, smart cities, and medical devices in hospital rooms connected to the nurse station. There are also many examples that solidly fit in the IoT vision: home thermostats, Internet webcams, emerging weather-based applications[12].
Summarized, for the purpose of this paper:

**CPS are** engineered systems that are built from, and depend upon, the seamless integration of computational algorithms and physical components. They may be built using IOT technology, and use the Internet, but other networking or communications mechanisms are possible.

**IV. CPS/ IOT Reference Architectures and Models**

Just as the definitions of CPS and IOT are not totally agreed upon, the reference architectures are not fully defined. CPS architectures may vary significantly based upon the problem domain, such that a general purpose one may not be useful [13]. Nevertheless, the need for common languages and understanding is important, and the figure below is a high-level IOT Reference Model, created by the Architecture Working Group of the IoT Forum Steering Committee (which involved 128 companies)[14]. This model is shown below in Figure 2 and Figure 3. The first slide gives an overview of the 7-layered stack; the second provides additional insight as to the different attributes and objectives of different layers of the model.

![IoT World Forum Reference Model](image-url)
Although both the CPS and IOT architectures and models are still in flux, will likely continue to evolve, we believe the IOT reference models are strongly applicable and useful for CPS. For purposes of this paper, we will use the IOT reference model shown above, and consider the IOT as the enabling, underlying technology required to implement a CPS.

V. Competencies for the CPS Software Engineer

Our goal of this work is to first present the new B.E. in software engineering, which is unique in its breadth of general engineering and depth in software engineering. Next we consider how well this program fits the needs for a CPS-focused software engineer. Our approach is to obtain a high-level, objective, evidence-based preliminary proposal for additional the CPS competencies required and then compare it to the curriculum. In order to understand these competences, we analyzed the IOT Reference Model and Definitions (above), the NSF-sponsored CPS Curriculum Recommendations and the general area of Embedded System Design.

In our reading, the IOT/CPS reference models imply that the ideal IOT undergraduate Software Engineer would have a working understanding of all levels of the reference model, more detailed and hand-on abilities in levels 2, 3, 5, and 7 of Figure 2, and have built simple CPS systems utilizing at least 4 levels of this IOT stack, including the data analysis and application levels as well as the embedded systems levels. It seems crucial to cross the intermediate levels of the stack, having systems that span both the real-time and non-real-time aspects of CPS.

NSF has funded the National Academies Committee on 21st Century Cyber-Physical Systems Education. The current CPS curriculum recommendations are only available in as an interim report [17]. Consequently, the Stevens’ curriculum was reviewed with James Sturges who is...
committee co-chair. Mr. Sturges observed that the SWE B.E. could be considered a model for an undergraduate CPS Engineering Curriculum [18].

Finally, it is important to not lose sight that CPS systems are minimally based on simple embedded systems, which many traditionally-educated software engineers do not experience nor understand. CPS Software Engineers need to be able to design and build simple embedded systems. They need to understand physical dynamics, operating systems, and real-time systems programming.

One potential conclusion could be that a CPS SWE should have all the competencies of both computer engineering (CE) and software engineering, with data analytics, high level systems engineering, and interdisciplinary skills added in. This is impractical for all but a few exceptional people.

If we prioritize the knowledge instead, and consider roles and responsibilities on projects, it may be that the CPS SWE should NOT duplicate the CEs knowledge expertise. Instead, one could focus on the software-intensive portions that are more uniquely in the software engineering realm. That is, have a broad knowledge of CPS and IOT with depth in a few areas that require both software engineering and CPS/IOT competence. Utilizing this principle, we would see CPS SWEs concentrating on bridging the physical and cyber world, which would be primarily focusing on levels 3-5. As they grow to become system architects, they would require more depth in understanding the interaction between all of the layers, and the overall understanding of all of the components, but again, focusing on the AND between cyber and physical. Depth in other areas such as data analytics, networking, or embedded systems would be useful, but not required.

Based upon this line of reasoning, we hypothesize that the ideal CPS Software Engineers require the areas of expertise in Figure 4 in addition to that which a traditionally educated SWE would possess, with a relative level of knowledge (the scale is 1 to 5, with 5 being high)

<table>
<thead>
<tr>
<th>Knowledge Area</th>
<th>Recommended Level of Knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Filtering and Storage</td>
<td>5</td>
</tr>
<tr>
<td>CPS/IOT Technologies/ Architectures</td>
<td>5</td>
</tr>
<tr>
<td>Interdisciplinary Skills</td>
<td>5</td>
</tr>
<tr>
<td>Systems Engineering and Architecture</td>
<td>3</td>
</tr>
<tr>
<td>Command and Control systems architecture and design</td>
<td>3</td>
</tr>
<tr>
<td>Embedded Systems</td>
<td>2</td>
</tr>
<tr>
<td>Physical and Fluid Dynamics</td>
<td>2</td>
</tr>
<tr>
<td>Data Analytics</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 4

---

3 Mr. Sturges also suggested that with the 141 required hours, the program might be considered a masters program.
VI. Comparison of CPS SWE to Stevens Undergraduate B.E. in Software Engineering

We compared our hypothesized level of CPS knowledge to the coverage we expect to be provided in the existing SWE program, as shown below in Figure 5. We have two columns of expected coverage. The first assumes the undergraduate SWE curriculum, but an unspecified senior design project. As indicated by the column (and by the prevalence of red and orange), there are significant gaps. The third column in the chart shows the expected coverage assuming a well-structured CPS senior design project. Although there are still gaps, the coverage is improved significantly. The potential for targeted co-op experiences could possibly further close the gaps and increase the coverage.

<table>
<thead>
<tr>
<th>Knowledge Areas</th>
<th>Proposed Level of Knowledge</th>
<th>Coverage in SWE BE program</th>
<th>Coverage with CPS Senior Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Filtering and Storage</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>CPS/IOT Technologies/Architectures</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Inter-disciplinary Skills</td>
<td>5</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Systems Engineering and Architecture</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Command and Control Arch &amp; Design</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Embedded Systems</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Physical and Fluid Dynamics</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Data Analytics</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 5

Based upon this analysis it appears that Stevens’ current SWE B.E. curriculum coupled with a CPS focused senior design project may be a reasonably good model program for a CPS software engineer program. Figure 5 highlights areas that could be strengthened and we expect to add additional technical material to the CPS program.

VII. Additional Work on CPS Software Engineering Curriculum

The consideration of the SWE B.E. program as a CPS SWE program goes beyond the original intent of the new B.E. program. Although our initial analysis and initial feedback is positive, we are continuing to seek feedback from the community and will continue to assess the results as our students progress through the program. If sufficient community interest is found we will propose a follow-up workshop on this topic.
VIII. Summary

Stevens’ roots are grounded in a multi-disciplinary approach to engineering education, beginning with founding of the university in 1870 around a singular “Mechanical Engineer” degree, to the current environment where all engineering majors have broad and common courses leading to the “Stevens Engineer” title.

This paper describes a new software engineering B.E. program at Stevens that will produce Software Engineers in the Stevens’ Tradition. It is unique in the breadth of engineering and the depth in software engineering.

We have compared this program against our view of the skills and competencies required by CPS/IOT software engineers, and found it a good match. We believe this approach will be necessary for future CPS/IOT engineers who will need to know a lot of software and bits of mechanical, civil, electrical engineering as they integrate their solutions into the real world. We have also identified several areas where we can extend the educational material to further strengthen this program for CPS/IOT. Given that we have just taken the first step of launching the program in the fall of 2015, there will subsequent papers analyzing the student outcomes.

IX. Acknowledgements

This visionary for this program is the provost of Stevens, George Korfiatis, who asked for a software engineering program embedded into the undergraduate engineering program.

Our software engineering curriculum expert is Mark Ardis, a self-described curriculum geek, who has contributed so much in so many ways to Software Engineering curricula worldwide. He is the true designer of this curriculum.

Our Dean, Dinesh Verma, has been steadfast in his vision of the need for a program for CPS. We appreciate his prescience.

X. References

[18] Sturgis, J., private conversation, November 2015