The Hatchery: An Agile and Effective Curricular Innovation for Transforming Undergraduate Education

Tim Andersen
Boise State University
tandersen@
boisestate.ediu

Amit Jain
Boise State University
ajain@
boisestate.edu

Noah Salzman
Boise State University
noahsalzman@
boisestate.edu

Don Winiecki
Boise State University
donwiniecki@
boisestate.edu

Carl Siebert
Boise State University
carlsiebert@
boisestate.edu

Abstract

The Computer Science Professionals Hatchery utilizes strong partnerships with industry and a vertically integrated curriculum structure, embedding principles of ethics and social justice and diversity, to create a nurturing, software company environment for students that also provides tools to allow them to take on the challenges of real-life company environment. The goal is to produce graduates who are well-rounded, who have a shorter pathway to full productivity after graduation, who can be leaders, and who can operate as agents of positive change in the companies where they work.

1. The CS Professionals Hatchery

The Computer Science Professionals (CSP) Hatchery seeks to transform undergraduate education in Computer Science by replicating the best elements of a software company environment, layering in moral, ethical, and social threads with entrepreneurship and professional skills. The goal is to create a curriculum and environment that produces graduates with the experience, training, and skills necessary to swiftly integrate into software company workflow and influence culture, shortening the path from graduation to being productive and beneficial. While this paper focuses on Computer Science Education, we believe that the Hatchery structure can be adapted to improve student outcomes in any subject area.

Computer science curriculum often focuses on technical aspects while relegating ethics to a single course. Issues of inclusivity and teamwork aren't integrated into the curriculum so cultural problems in the profession continue to be propagated. Industry complains about a lack of responsiveness to rapidly changing technologies, and a corresponding lack of real-world relevance in the curriculum – i.e. students may learn the theory but current technologies and practice are not sufficiently integrated into the curriculum. The CSP Hatchery is an attempt to address all of these problems.

The CSP Hatchery utilizes a progressive academic curriculum structure where students at all grade levels work with each other. This structure focuses on three curricular innovations: (1) Infusion of ETHICS AND SOCIAL JUSTICE principles, starting at the first course taken by Freshmen CS majors and continuing throughout the curriculum. Our goal is to inseparably infuse ethical/moral elements into the practice of software engineering for our students, to empower our students to be agents of revolutionary change in reshaping the practice of computer science to be a more just and inclusive profession. (2) Short, narrowly focused, agile courses, which we call HATCHERY UNITS, are threaded with regular course work and are used to infuse foundational concepts and skills at key points into the curriculum. Industry involvement in the design and delivery of hatchery courses ensures that they focus on the skills and capabilities most useful to students in the work that they will actually perform in an industry setting. (3) Vertically Integrated Teaching and Learning (VITaL) curriculum. Instead of being in siloes, students at all grade levels work with and learn from each other on industry-sponsored projects, fostering a strong sense of community amongst students, faculty, and industry.

The CSP Hatchery project is currently in the third year of its implementation, with two years remaining. Since the start of the project in Fall 2016, five required and three elective Hatchery courses have been designed and offered. Infusion of ethics and morality and vertical integration is also in the process of implementation.

2. Related Work

Over time, there have been efforts to address matters of ethics and social-justice in techno-scientific fields. Historically, most of these have focused on the former through the post-hoc analysis of engineering failures from a mostly technical perspective ([30]). More recently there has been considerable effort to develop more nuanced, philosophically-oriented approaches, or behavioral-psychology approaches to ethics in a society all-but built around techno-science ([23], [30], [31], [32], [33]), and even more focus on trying to

URI: https://hdl.handle.net/10125/60216
ISBN: 978-0-9981331-2-6

understand how to understand the problem itself in a world increasingly dependent on full-time access to technologies that themselves reflect ethical dilemmas in our society ([34], [35], [36], [37], [38]).

However, it is worth noting that efforts to actually introduce these issues into curricula appear to have usually accepted the traditional approach of concentrating all such content into one course. While this may be easier to accommodate from an administrative angle (and one cannot deny the substantive pressure against innovation in the structuring of components in a degree plan in the very bureaucratic world of higher education), the result is that while students may be required to complete an 'ethics course' as part of their education and degree completion, they have not been provided with examples or strategies for actually incorporating this content into their day to day practice as computer scientists and engineers. This is the case even if the content of that course went beyond the usual issues of professional and legal responsibility, copyright, contract considerations, etc.

With this in mind, and following the idiom of 'regular practice, distributed practice,' and the use of the methodology of cognitive apprenticeships from educational psychology ([24], [39], [40]), the CSP-Hatchery aims to incorporate content related to ethics, professional morality and social justice across the undergraduate curriculum through both 1-credit 'hatchery unit' courses, and by partnering with technical faculty to develop instructional modules that fit professional ethics into otherwise 'purely technical' courses. In this process, students will have many opportunities to puzzle with and apply structured processes for addressing ethical and social justice issues within the context of computer science practice and product development, and thus graduate better prepared for addressing these issues in their real-world practice.

The idea of short, agile Hatchery courses is novel. Several programs do offer 1-credit supplementary courses but we are not aware of any program using them in a foundational way like we are doing.

Vertical integration isn't a new concept in curriculum reform. See [25], [26], and [27] for examples in computer science programs. However, vertical integration of technical, social and ethical issues is a novel application. Instead of being concentrated in one or two courses, we are threading these concepts through the curriculum using multiple courses at various academic levels.

3. Hatchery Units

Hatchery Unit (HU) courses are envisioned as light-weight (generally 1 credit hour or less), industry inspired, focused courses addressing key skills and core concepts, such as foundational values (like teamwork, inclusivity, ethical frameworks), navigating computer systems (expert navigation in a system, systems administration, scripting to automate tasks, etc.), security, version control, agile development, and intro to databases, which are important for students to know in order to be successful both in our program and in their internships/jobs. In some cases, HUs help to 'level the playing field' by providing students without extensive CS experience integrate more readily into the undergraduate curriculum and become more competitive for professional internships. HUs are delivered over a short time-frame, such as 5 weeks or 7 weeks, enabling students to take multiple HUs back-to-back in a single 15-week semester if they so desire.

HUs prime students with the core knowledge they need in focused skill areas at specific, key points in the curriculum. The focused content delivered in the HU is then woven through the regular full-semester courses in the curriculum from the point of the Hatchery Unit onward, with subsequent courses incorporating and continuing to exercise HU skill sets through additional course content, activities, and assignments. Students in HU courses work with and learn from faculty, industry professionals, senior capstone teams, and from each other. Industry professionals are brought in to assist with HU content delivery as appropriate.

The requirements were gathered via a group of 17 industry representatives who responded to an inquiry as to the Knowledge, Skills, and Abilities (KSA) their company looks for when hiring. These KSA were collected from the individuals (or groups within a company) and collated into unique KSAs. The KSAs were then grouped into 6 unique categories which emerged as the KSAs were collected and analyzed - these include (Technical, Professional, Collaboration & Teams, Research & Development, Entrepreneurship, and Business). The industry representatives were then pulled together in a meeting in which they voted for the two KSAs in each group that were most important to them. The votes were tallied, reviewed, and used as the basis for the creation of new 1-credit Hatchery Unit courses, enhancements to existing CS course content, and threading the content into additional CS courses.

3.1. HU Integration into Current Curriculum

Five required HU courses have been integrated into the current Boise State Computer Science curriculum along with several elective HUs as well. Figure 1 shows how these required HUs (orange shaded rectangles) integrate with regular course work. The course catalog descriptions for these courses can be found at the CS-HU website [29].

The *Foundational Values* HU (see Section 3.1.1 for details) and subsequent team activities in follow-on courses sensitize students and give them the social and professional-skills they need to be more effective and inclusive members of software development teams. The *Agile Development*, *Navigating Computer Systems*, *Intro to Database System Usage*, and *Version Control* HUs add valuable technical knowledge and skills that students previously did not have until later in the curriculum (or often as a side topic in other courses) and that help students hit the ground running in their internships with our industry partners.

Typically, software/tech companies have required students to take data structures (CS 321) before they will consider hiring them for internships.

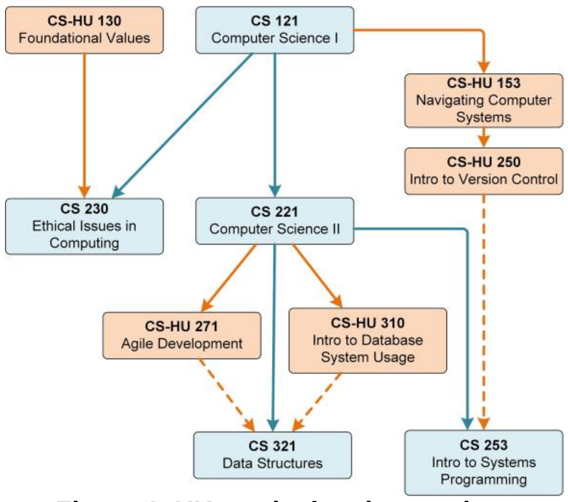

Figure 1. HU curriculum integration.

With the addition of the five HU courses, students who have taken the data structures course now have several additional professional and technical skills that make them much more capable and able to integrate into company projects and workflow as interns. The HU structure makes it possible to introduce these important concepts into the curriculum with minimal overhead and maximum benefit for the students.

Table 1 gives the number of students who have taken each of these required courses so far.

Table 1. HU student enrollments.

HU Course	Start	#Students
Foundational Values	Fa'17	232
Agile Development	Fa'17	52
Navigating Computer Systems	Sp'18	182
Intro to Database System Usage	Sp'18	42
Version Control	Su'18	15

We have also added several elective 1 credit HU courses that allow students to explore other relevant topics. These elective HU courses, shown in Figure 2, include courses focused on Human Computer Interaction; Software Testing; Secure Programming; and Technical Interviews, Jobs and Careers.

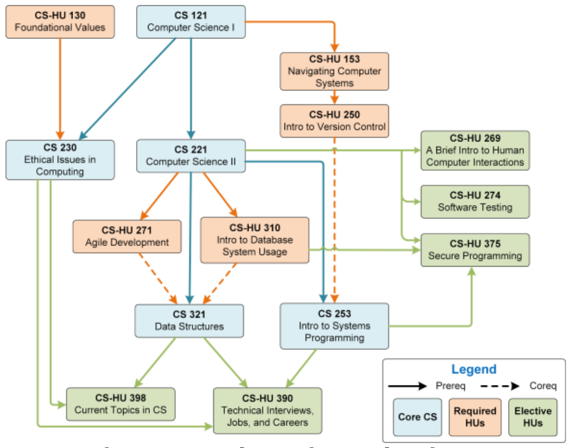

Figure 2. Selected HU electives

CS-HU 390 *Technical Interviews, Jobs and Careers* provides an example of how hatchery units can help students level the playing field and increase their readiness for computing careers. This course teaches students the technical interview process to start with but then leads them to investigate what their first job and then their career can be like. Fifteen industry professionals participated in the first offering of the course, helping with invited lectures, mock interviews and panel discussions. A significant part of the course is to encourage and support underrepresented students by demystifying the interview process.

3.2. Ethics and Social Justice

One need look no further than the headlines of major newspapers and online reporting to find breaches of social justice that adversely affect underrepresented groups in CS professions and in the commercial use of CS products ([1], [2], [3], [4], [15], [16]). Academic research has long focused on issues of bias in society. With new focus on STEM industries and even academic practice, we now know more clearly than ever how widespread and deeply rooted are these biases ([7], [8], [9], [10], [11], [12], [13], [14], [20]. We can no longer assume that computer science is simply meritocratic and that those who do not succeed are somehow inherently incapable. Rather we have to face the fact that embedded bias prevents inclusion and diversity in the field, limiting the available talent pool.

It is with this backdrop that we saw it necessary to institute a new beginning course for computer science students. The first course in our curriculum is *CS-HU 130 Foundational Values*. CS-HU 130 takes a path different from most courses in `computer science and engineering ethics` that review well known disasters of poor design or poor planning, and ask students to apply formal ethical theories to an academic (i.e., abstracted and detached) analysis of their conditions. CS-HU 130 is designed as a problem-based learning experience in which students (a) review case studies in which bias is reflected in the context of actual computer-science related work ([16], [3], [5]), and in the design and application of computer-science products that reinforce that bias and loss of social justice ([1], [2]), and then (b) in teams, work to apply a problem-analysis and problem-solving rubric based on Rawls' Theory of Justice ([17]) and principles of organizational performance improvement to draft proposed solutions that can be enacted both within computer science and more broadly in organizations and in society itself.

Additionally, these problem-based learning teams use a research- and practice-based rubric for scoring their teammates' contributions ([6], [21]), to assess teammates' contributions to the team product, and their own motivation to contribute to the team's interactions. The curriculum of CS-HU 130 is designed to guide students to assess what happens `out there` in problematic case studies, what is happening in their own problem-based learning teams, and if problems are identified to propose actionable solutions.

Some students are excited by this curriculum, providing feedback that it has altered their perspectives, and in some cases even increased their interest in computer science as a field in which they can contribute lasting positive change. One student said, "…my parents were surprised when I talked about [bias toward underrepresented groups in CS] when I went home for Thanksgiving. My Dad suggested that I should talk to my high-school CS teacher and ask if he would be interested in learning more about these things." Another student who was debating whether he should major in computer science or philosophy and chose CS because of future job prospects, said, "…I'm really glad I chose CS, because now I know I can do *both* CS and ethics!" A third student described how one of the topics in CS-HU 130 convinced her she should focus on artificial intelligence and machine learning: "…when I saw that software biased against minorities in things like facial recognition and voice recognition, it convinced me that I had to focus on that. I am mixed race and speak English as a second language." A female student from one section of the course asked for extra readings and research articles on the topic of the equality of women and men in math and science knowledge and skill. She said, "…when you told us about research that said women were as good as men in math, it made me feel, like, 'Yeah!' – now I know that I'm not weird just because I like math and I'm good at it."

Additionally, over the eight sections of CS-HU 130 offered in the 2017-18 academic year, student teams generally improved the depth and breadth of their solutions to problem-based learning cases through the five-week course, showing an improvement in curriculum-related knowledge and skill. As reflected in the quotes included above, in interviews with students following CS-HU 130 they sometimes reference case examples used in that class before describing episodes from personal experience in which circumstances may expose bias against others. This suggests the CS-HU 130 curriculum serves as the basis for a new understanding of factors related to inclusion, diversity and social justice, especially how it relates to professional computer science contexts and how they are already realizing new possibilities and new potentials for themselves. Regular interviews with these students starting from CS-HU 130 though their subsequent years in the CS curriculum aims to track such things in detail, to identify places where (or if) students are applying what they have learned in in CS-HU 130 in other courses or other areas of their lives.

We acknowledge that one course, taken in the first semester of a student's career is only a small step, and that is why the Hatchery concept requires follow-on courses to incorporate learning experiences that focus on similar issues specific to the technical focus of those courses. For example, the CS-HU 153 (Navigating Computer Systems) course has a module where the students have to apply foundational concepts to challenging social and ethical issues related to systems. They are provided with two scenarios involving ethical dilemmas concerning systems that were drawn from actual industry events. Their assignment is to identify the stakeholders, their interests, concerns and risks, and then apply one of the five ethical theories (Utility, Rights, Justice, Common Good, and Virtue, See [22] for more information on these theories) to analyze the situation.

This is one out of the six total modules in a technical course but it ties technical concepts with the social and ethical dilemmas that they can lead to. These concepts are also being integrated into other HU courses and core CS courses. In this way, the Hatchery

Unit concept aims to reinforce issues and practices countering bias and breeches in social justice throughout the computer science curriculum.

Additionally, other faculty have begun to express interest in adapting their curricula to incorporate these topics with the assistance of faculty from CS-HU courses. These include instructors for the Senior Capstone course, who will be incorporating some of the instructional content and evaluation tools introduced in CS-HU 130 in order to put emphasis on professional skills within project teams, and instructors in data science and machine learning courses are now including case studies of unintended bias in the products of these technologies. We are investigating ways of allowing students in CS-HU 130 to participate as 'consultants' to project teams in other courses.

These outgrowths serve to further embed Hatchery concepts across the curriculum, deepen the implementation of VITaL across courses, and expose another avenue through which to realize the overall goals of this project.

3.3. Advantages of Hatchery Units for Faculty Development

HU courses are intentionally lean, enabling these courses to be quickly designed and incorporated into the curriculum. They are intended to foster a much more agile and adaptable curriculum that is more aligned with industry needs and that can keep pace with the rapidly changing software engineering landscape. While not required, for HU courses it is encouraged that at least some of the course content be online (and for some HUs almost all of the course content is delivered online). The idea is to identify core knowledge areas within the curriculum and use HU style courses that are easy to pick up and teach in order to deliver that core knowledge to students. This makes it easier for both faculty and industry professionals to create these courses and deliver them.

Offering HUs partially or entirely online also increases flexibility for offering courses — allowing more courses to be offered than would otherwise be allowed by physical classroom space. This also benefits transfer students by providing added flexibility for them to complete courses they could not have gotten in previous institutions.

HUs have other advantages from a faculty development perspective. For required HUs we generally teach multiple sections of the HU in a single 15-week semester. These sections can be taught back-to-back in two or three 5-week sessions, or concurrently in the same 5-week session.

Research-active faculty are required to teach at least 3 credit hours' worth of courses per semester, and to fulfill this requirement they may choose to teach either three 1 credit hour HUs, or one 3 credit hour regular course. For new faculty, teaching the same HU course back-to-back allows them to receive course feedback and implement course improvements up to two times in a single semester, a significant reduction in the performance/feedback loop that approximates an agile development process, and which should lead to faster teaching performance improvement. Also, teaching a HU course back-to-back three times in a single semester is much easier than teaching a single 3 credit hour course due to the reduced course prep time, which frees faculty time for their research and other responsibilities. Additionally, faculty may choose to teach all three sections of a HU concurrently, leaving them completely free to do research during the remaining 7-10 weeks of the semester.

We have also created other incentives to increase HU participation. Faculty designing a new HU course get extra summer salary or release time. To encourage faculty to rotate through multiple HU courses, the departmental workload policy counts two HU courses the same as three HU courses when a faculty teaches a new HU course.

3.4. Assessing the Impact of the CSP Hatchery

As we are still relatively early in the implementation of the CSP Hatchery Project, we currently have limited data establishing the effectiveness of this approach. Moving forward, we will utilize several key performance indicators (KPI) to assess the success and impact of the CSP Hatchery approach. Primary among these KPIs are the assignments that we have tied to assessment of ABET outcomes, which we have consistently collected as part of the accreditation process. These include assessments in Data Structures (CS 321) and Intro to Systems Programming (CS 253) as well as other courses down the pipeline. Four of the new HU courses are pre/co-requisites for existing courses and we will be compare the historical data to new data after students have gone through the HU courses. In particular, we anticipate that the increased focus on teamwork and project management infused through the Hatchery Units will improve student performance on assessments designed to measure these outcomes.

We are also using student records and enrollment data to measure the effects of changes. These include number of HUs offered and number of students enrolled (Figure 3), along with tracking retention rates and other enrollment data with a particular focus on women and underrepresented minorities.

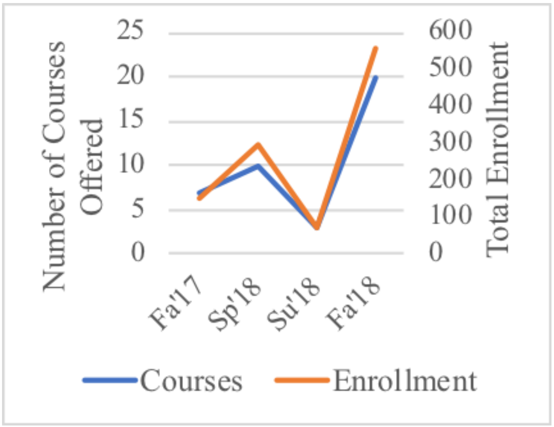

Figure 3. HU Offerings and Enrollment.

Feedback from industry is another important component as the new students interview and are placed. We have already received positive feedback from industry about students who are going through the CSP Hatchery.

Other more novel approaches to assessing the impact of the CSP Hatchery project will include interviews and focus groups with students and faculty, and utilizing sociograms and social network analysis to explore how students and faculty build connections and community within the undergraduate CS program at Boise State University.

4. Importance of Industry Involvement

Developing and maintaining strong industry partnerships is critically important for the development of the software company environment that is envisioned for the CSP Hatchery. Without strong industry relationships, it is difficult to know about the issues that industry faces, and the current trends in industry in terms of tool usage and desired skill sets, and it is difficult to get the real-world feedback on graduate performance that is a necessity for maintaining a relevant and targeted curriculum.

Good industry relationships are also required to be aware of the best practices amongst industry partners, and in order to design customized methods for identifying and addressing moral and ethical issues relative to professionals in the workplace in computer science.

Having a mutually beneficial relationship with industry partners requires academic departments to create, foster, and disseminate a value-proposition that is enticing to them. This value proposition can certainly appeal to altruistic desires to be a “good citizen” and give back by providing benefit to the program and students, but could also appeal to industry needs, such as having a talent pool that is well-trained and fits industry’s desired skills and abilities, as well as giving those industry partners who are actively benefiting and participating in program improvement an inside track to this talent pool. The key is to understand what motivates each industry partner and speak to that motivation if feasible. In cultivating these relationships, it is extremely important that industry feels that their feedback and concerns are being heard and actively addressed.

Well before we applied for the RED program, we began cultivating industry relationships and feedback through one-on-one contacts and relationships, and through invited membership of high-level industry representatives on our industry advisory board, which meets twice a year. For several years, feedback from our industry partners has been actively incorporated into curriculum changes and design, and progress reports have been duly reported to industry on a regular basis.

In 2014 we also established a scholarship/internship program, called *Expand.CS*, funded by industry donations, which to date has generated over $534,000 in industry funded scholarships for 60 students who have also participated in over 40 internships at different companies. Industry partners who donate money to the *Expand.CS* program meet with faculty to assist in reviewing student application materials and awarding scholarships, and are given an inside track to hiring these students as interns. Through these and other activities we have developed a reputation for responsiveness to industry needs, and quality graduates, which made it much easier to ask for and receive their input and help on our NSF funded CSP Hatchery project.

In conceptualizing the CSP Hatchery, we wanted to ensure and ease industry participation in both the design and the offering of curriculum elements, and this was one of the factors considered, and advantages of, the Hatchery structure. It is much easier for industry partners to commit to helping in an accelerated (shorter) course vs. assisting in a regular 3 credit hour course for an entire semester. It is also easier and more motivating for them to take on the task of assisting in the design of a focused topic course that directly matches a clear need for them. The HU course concept lowers the bar for the participation of industry professionals.

Upon receiving word that the grant would likely be funded, we contacted industry partners and asked them to brainstorm on the skills and abilities that are important for success but that are typically lacking in CS graduates. Each industry partner independently put together a team to do this, and we collected and summarized the results of this effort to reduce overlap. We then met with the industry partners together to discuss and prioritize their feedback. This was then taken to the faculty, and over the course of six months faculty worked on how to address the prioritized industry feedback, and curriculum changes were proposed and designed. Another meeting was called with industry partners and the new courses and curriculum design was presented and enthusiastically approved.

A total of forty industry professionals ranging from junior engineers to senior executives from twelve different companies have participated in the CSP Hatchery project so far. The companies range from large multinational technology companies to smaller, local software companies. It also includes non-technology companies from other areas that have a strong interest in software solutions to their problems. Their ongoing participation in the project allows us to incrementally refine and steer our efforts toward providing a curriculum that meets the technical and social needs of the industry.

5. Vertical Integration

The Vertically Integrated Teaching and Learning (VITaL) curriculum is vertically integrated in two ways:

1) Vertical threading of HU course concepts through HU and regular courses. HU courses introduce students to core knowledge areas and give students preliminary exposure and experience in these areas. The students are then required to exercise the principles/skills that they have learned in the HU course in follow-on courses. This requires a high level of coordination between courses (and the faculty teaching them) to ensure that students are given multiple opportunities to learn and apply core concepts.

2) Vertical integration of student teams on capstone projects. The core skill formation activity in VITaL HU curriculum design involves HU student teams working with senior capstone teams on their capstone projects. Specifically, the knowledge taught in HU courses will be leveraged to create HU student teams that work with the senior capstone teams on some aspect of their capstone project related to the skill that the HU is delivering. In effect, students in HU courses act as a sort of subject-matter consultant to the senior capstone teams.

At the same time, since capstone teams are formed of senior level students who have already gone through this process, they are prepared to perform as mentors for the HU students they are working with, to help HU students deepen their knowledge of the systems in which particular skills are applied. In their performance in the mentoring role, the core concepts will be reinforced for these senior level students, and they will form beneficial relationships with juniors, sophomores, and freshman.

6. Building Community

Building community to create a more welcoming environment for students, especially those historically underrepresented in undergraduate computer science programs, is another overarching goal of this project. Grounded in Wenger's ([19]) theory of Communities of Practice, we are exploring changes to the curriculum and structure of our program that will build community among students, faculty, and industry partners. This goal is embedded across multiple elements of the Hatchery curriculum, including the focus on ethics and social justice, the development of Hatchery Units, and building the VITaL curriculum. As described in the previous section on Ethics and Social Justice, computer science and software development environments can often be hostile to women and underrepresented minority students, making it difficult for members of these groups to develop a sense of community or belonging in their computer science degree program. By helping all students to become more aware of these issues, we hope to reduce bias, which should in turn help to build a more welcoming community for all students.

The nature of the Hatchery Units also promotes building community among faculty, students, and industry. Faculty design and implement all Hatchery Units as part of an instructional team, strengthening the faculty community and creating opportunities for faculty to learn from each other regarding their teaching practices. VITaL curriculum design that involves threading of HUs with other HUs and normal courses also promotes faculty community as they have to work more closely together. Many Hatchery Units were developed in response to industry needs and input, and often involve an industry partner as part of the course development team. This creates opportunities for further collaboration with industry partners, and helps to integrate faculty and students in the local software development community. Hatchery Units also allow faculty members an efficient way to develop a new course related to their research programs, creating an opportunity for training and recruiting students to work in their research groups, which creates another entry point for building community within the department.

Implementing the VITaL curriculum also represents a novel way of building community in an undergraduate computer science program. Most students tend to take classes with the same group of peers progressing through the curriculum at the same time. While this does create a sense of community within a given class year, it minimizes students' opportunities to interact and build community across grade levels. The VITaL curriculum transforms this paradigm by having students across all grades working together on shared design projects, allowing students to work with and get to know peers at different points in the curriculum. Through these interactions, students will both build community *across* grade levels and learn more about the experiences of students further along in the curriculum, which may better prepare them for their future classes and help students to persist in their degree program. Overall, the Hatchery structure is designed to create a more nurturing environment for students, and building community is an important and intentionally designed aspect of this transformative approach to undergraduate education.

7. Industry Impacts

The industry partner involvement in the CSP Hatchery project, explained earlier, shows a comprehensive approach to engagement. Even though the project is only starting year three of the five-year commitment, evidence of positive benefits to the software and information technology industry are already recognized. As part of the Outside Evaluator's oversight of project activities and effects, the Outside Evaluator interviewed eleven industry partners on their beliefs and perceptions on the CSP Hatchery project, the preparation of students for employment, and social skills/diversity in the work environment as well as in their organization. Interviews lasted between 30 to 45 minutes and followed a protocol that directed the recorded discussions. An outside firm transcribed the recordings to avoid any transcription bias.

An analysis of the interview data show that industry partners view the CSP Hatchery project as a commendable effort on the part of the Boise State CS Department. For example, one partner stated "I think the Hatchery approach is probably one of the biggest strengths … 'we' hope it stays," Partners do recognize that the project is early in the effort to graduate a more well-rounded student, but proclaim that the project is well on its way to achieving this goal. In addition, the industry partners believe the CS Department does an excellent job with encouraging and facilitating industry engagement in curriculum activities and with providing early access to students who will enter the workforce. The Outside Evaluation will seek feedback from industry partners two additional times in the coming years to fully identify the impact of the CSP Hatchery project through the eyes of the industry partners.

8. Challenges

Complexity of the curriculum changes requires careful attention to details such as the timing of the introduction of HU courses, making sure options exist for students "caught in the middle of the transformation," scheduling of courses, and proper communication to the students. For example, we had originally planned on updating the requirement of new Hatchery Unit prerequisites for the Data Structures course (CS 321) for Spring'18 but we pushed it back to Fall'18 to allow students caught in the middle one more semester to complete the old version of the course, extending the original one year notice to one and half years, which was sufficient to resolve almost all of the concerns.

Advising complexity needs to be addressed as well. We worked closely with the college advisors so they are aware of the changes and can advise students on what they can take advantage of and how. For example, many juniors and seniors don't need HU courses as they are on the older catalog, but we are allowing them to take HU courses in place of one upper-division elective. We have held a special workshop for the advisors and we pay attention to the "word on the street" that we get from them. We have created a website especially for students (also used by advisors and faculty) that acts as a reference.

VITal curriculum has serious logistics challenges. How do we get freshmen and sophomores to work together with juniors and senior in a meaningful way without having scheduling nightmares? We are reviewing several possible approaches to make this feasible. These approaches will be shared at large so others who want to implement a VITal curriculum can benefit from our solutions.

Scheduling Hatchery Unit courses such that they can be taken consecutively rather than concurrently is important in keeping a balanced workload for the students. However, scheduling them in first/second five (or seven weeks) is also important as standard 3 credit courses tend to ramp up towards the end.

Finding instructors from industry has been relatively easy due to the strong relationships and connections that we have developed over time. We also incentivize industry involvement by paying industry partners for their part in both the development and delivery of HU-courses, and we always pair the industry partners with faculty coordinators so they have proper support. Currently, grant funds are used to supplement industry pay. So, when the grant is over a challenge will be to find money in the department budget to continue this model.

Another challenge is getting faculty buy in. Initially, the grant is being used to provide summer salary or release time to faculty that wrote proposals to create HU courses. This has been successful in getting the courses off the ground. The next challenge was how to update the workload policy to ensure that the HU courses count as appropriate amount of workload. Faculty were very concerned about this issue, which we resolved with an updated workload policy. Two 1-credit HU courses count the same as one 3-credit traditional course when a faculty teaches the HU course for the first time. This creates an incentive for faculty rotate through multiple HU courses. The rotation through HUs also helps faculty develop a more comprehensive understanding of the curriculum. Once the workload policy was updated, that resolved most of the concerns faculty had about the effect of the HU courses on their workload.

9. Conclusions

In this paper, we have described the design and initial implementation of the *Hatchery*: an agile and novel curricular innovation that has the potential to transform undergraduate curriculum not only in computer science but other areas as well.

The starting premise of the Hatchery is to introduce short accelerated courses and vertically integrated opportunities to develop professional skills in students. Close collaboration with motivated industry partners in the design and delivery ensures the relevance of the Hatchery. This also increases the motivation and interest from the students.

By introducing students to social, moral, and ethical foundational values from the start and threading them through technical courses, we can create agents of change that can go out into industry and create lasting improvement in the culture of the companies and beyond.

The Hatchery model can also benefit faculty development due to the requirement for more threading between courses. The scheduling structure has the potential to help faculty more quickly improve their teaching performance, while simultaneously giving them more time to perform research.

The Hatchery curriculum is structured to enable industry participation, and to enable adaptability to rapidly changing industry needs. The focus on job skills motivates students and naturally leads to better student engagement and performance. Being responsive to and producing a product that is more aligned with industry needs also leads to more engaged industry partners. The CSP Hatchery thus fosters a mutually beneficial and self-reinforcing relationship between industry, faculty, and students. We believe that the general model of the *CS Professionals Hatchery* represents a revolutionary approach to undergraduate education with potential to be adopted at other institutions and adapted to other disciplines.

10. Acknowledgements

We would like to acknowledge Ernie Covelli and Dianxiang Xu for their help with this paper. This material is based upon work supported by the National Science Foundation under Grant 1623189.

10. References

[1] Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016, May 23). Machine Bias: There’s software used across the country to predict future criminals. And it’s biased against blacks. [ProPublica]. Retrieved May 24, 2016, from https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

[2] Buolamwini, J. (2017). *Gender Shades: Intersectional Phenotypic and Demographic Evaluation of Face Datasets and Gender Classifiers* (Master of Science). Massachusetts Institute of Technology, Cambridge, MA. Retrieved from https://www.media.mit.edu/publications/full-gender-shades-thesis-17/

[3] Fowler, S. (2017, February 19). Reflecting on one very, very strange year at Uber. Retrieved February from https://www.susanjfowler.com/blog/2017/2/19/reflecting-on-one-very-strange-year-at-uber

[4] Lohr, S. (2018, February 9). Facial Recognition is Accurate, if You’re a White Guy. *New York Times*. Retrieved from https://www.nytimes.com/2018/02/09/technology/facial-recognition-race-artificial-intelligence.html

[5] Mims, C. (2017, August 13). What the Google Controversy Misses: The Business Case for Diversity. *Wall Street Journal*.

[6] Strauss, V. (2017, December 20). The surprising thing Google learned about its employees -- and what it means for today’s students. *The Washington Post*. Retrieved from https://www.washingtonpost.com/news/answer-sheet/wp/2017/12/20/the-surprising-thing-google-learned-about-its-employees-and-what-it-means-for-todays-students/

[7] Bazerman, M., & Tenbrunsel, A. (2012). *Blind Spots: Why we fail to do what’s right and what to do about it*. Princeton, NJ: Princeton University Press.

[8] Ceci, S., Williams, W., & Barnett, S. (2009). Women’s Underrepresentation in Science: Sociocultural and Biological Considerations. Psychological Bulletin, 135(2), 218–261. https://doi.org/10.1037/a0014412

[9] Ceyer, S., Chisholm, S., Friedman, J., Hewitt, J., Hodges, K., Hopkins, N., …, Stubbe, J. (1999). *A Study on the Status of Women Faculty in Science at MIT* (Manuscript). Cambridge, MA: Massachusetts Institute of Technology. Retrieved from http://web.mit.edu/fnl/women/women.pdf

[10] Crowston, K., & Howison, J. (2005). The Social Structure of Free and Open Source Software Development. *First Monday*, *10*(2). Retrieved from http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1207/1127

[11] Hill, C., Corbett, C., & St. Rose, A. (2010). *Why So Few? Women in Science, Technology, Engineering, and Mathematics*. Washington, D. C.: American Association of University Women (AAUW). Retrieved from http://www.aauw.org/files/2013/02/Why-So-Few-Women-in-Science-Technology-Engineering-and-Mathematics.pdf

[12] Natanson, H. (2017, October 20). `A Sort of Everyday Struggle`. *The Harvard Crimson*. Retrieved from https://www.thecrimson.com/article/2017/10/20/everyday-struggle-women-math/

[13] Rattan, A., Steele, J., & Ambady, N. (2017). Identical applicant but different outcomes: The impact of gender versus race salience in hiring. *Group Processes & Intergroup Relations*, OnlineFirst. https://doi.org/10.1177/1368430217722035

[14] Tonso, K. (2007). *On the Outskirts of Engineering: Learning Identity, Gender, and Power via Engineering Practice*. The Netherlands: Sense Publishers.

[15] Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016, May 23). Machine Bias: There's software used across the country to predict future criminals. And it's biased against blacks. [ProPublica]. Retrieved May 24, 2016, from https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

[16] Damore, J. (2017, August 4). Google's Ideological Echo Chamber: How bias clouds our thinking about diversity and inclusion. Retrieved from https://assets.documentcloud.org/documents/3914586/Googles-Ideological-Echo-Chamber.pdf

[17] Rawls, J. (1999). *A Theory of Justice* (Revised edition). Belknap Press of Harvard University Press.

[18] Gilbert, T. (2007). *Human Competence: Engineering Worthy Performance*. Washington, D.C. Pfeiffer.

[19] Wenger, E. (1998). *Communities of practice: learning, meaning, and identity*. Cambridge, U.K. ; New York, N.Y: Cambridge University Press.

[20] Hyde, J., Lindberg, S., Linn, M., & Williams, C. (2008). Gender Similarities Characterize Math Performance. Science, 321(5888), 494–495. https://doi.org/10.1126/science.1160364

[21] Frenkel, S., Korczynski, M., Shire, K., & Tam, M. (1999). On the Front Line: Organization of Work in the Information Economy. Cornell University Press.

[22] Vallor, S. (2016). *Technology and the Virtues: A Philosophical Guide to a Future Worth Wanting*. Cambridge: Oxford University Press.

[23] Adams, C., & van Manen, M. (2017). Teaching Phenomenological Research and Writing. *Qualitative Health Research*, *27*(6), 780–791. https://doi.org/10.1177/1049732317698960

[24] Bergamin, J. (2017). Being-in-the-flow: expert coping as beyond both thought and automaticity. *Phenomenology and Cognitive Science*, *16*, 403–424. https://doi.org/10.1007/s11097-016-9463-1

[25] Abler R., Coyle E., DeMillo R., Hunter M., Ivey E. (2012) Team-Based Software/System Development in the Vertically-Integrated Projects (VIP) Program. In: Thaung K. (eds) Advanced Information Technology in Education. Advances in Intelligent and Soft Computing, vol 126. Springer, Berlin, Heidelberg.

[26] R. D. Parslow, Vertical integration in group learning, Proc. of the Eleventh SIGCSE Technical Symposium on Computer Science Education (SIGCSE'80), page 130

[27] M. Baxter, B. Byun, E.J. Coyle, T. Dang, T. Dwyer, I. Kim, C.-H. Lee, R. Llewallyn, and N. Sephus, "On Project-Based Learning through the Vertically Integrated Projects Program", Proceedings of the 41st Annual ASEE/IEEE Frontiers in Education Conference, Rapid City, SD, Oct. 12-15, 2011.

[28] CS Hatchery Unit website (2018). Retrieved from http://coen.boisestate.edu/cs/cs-hu/#HUDescription

[29] Jasanoff, S. (2016). *The Ethics of Invention: Technology and the Human Future*. W. W. Norton & Company.

[30] Bielby, J. (2015). Comparative Philosophies in Intercultural Information Ethics. *Confluence: Journal of World Philosophies*, *5*, 233–253.

[31] Brewis, D. (2017). Social Justice 'Lite'? Using Emotion for Moral Reasoning in Diversity Practice. *Gender, Work & Organization*, *24*(5), 519–532. https://doi.org/doi:10.1111/gwao.12171

[32] Etzioni, A., & Etzioni, O. (2017). Incorporating Ethics into Artificial Intelligence. *The Journal of Ethics*. https://doi.org/10.1007/s10892-017-9252-2

[33] Gintis, H. (2011). Behavioral Ethics. In E. Slingerland & E. Collard (Eds.), *Creating Consilience: Integrating the Sciences and the Humanities*. Oxford University Press.

[34] Floridi, L. (2013). Distributed Morality in an Information Society. *Science and Engineering Ethics*, *19*, 727–743. https://doi.org/10.1007/s11948-012-9413-4

[35] Haraway, D. (1993). Situated Knowledges: The Science Question in Feminism and the Privilege of Partial Perspective. In E. Keller & H. Longino (Eds.), *Feminism and Science*. Oxford University Press.

[36] Heersmink, R. (2017). Distributed Cognition and Distributed Morality: Agency, Artifacts and Systems. *Science and Engineering Ethics*, *23*, 431–448. https://doi.org/10.1007/s11948-016-9802-1

[37] Henslee, A., Murray, S., Olbricht, G., Ludlow, D., Hays, M., & Nelson, H. (2017). Assessing Freshman Engineering Students' Understanding of Ethical Behavior. *Science and Engineering Ethics*, *23*, 287–304. https://doi.org/10.1007/s11948-016-9749-2

[38] Pennock, R., & O'Rourke, M. (2017). Developing a Scientific Virtue-Based Approach to Science Ethics Teaching. *Science and Engineering Ethics*, *23*, 243–262. https://doi.org/10.1007/s11948-016-9757-2

[39] Adams, C., & van Manen, M. (2017). Teaching Phenomenological Research and Writing. *Qualitative Health Research*, *27*(6), 780–791. https://doi.org/10.1177/1049732317698960

[40] Schön, D. (1987). Educating the Reflective Practitioner. San Francisco: Jossey-Bass.