A Cross-Curricular Approach to Fostering Innovation such as Virtual Reality Development through Student-Led Projects

Sherri Harms/John Hastings
Department of Computer Science & Information Technology (CSIT)
University of Nebraska at Kearney (UNK)
Kearney, NE USA
{harmssk, hastingsjd}@unk.edu

***Abstract*— As the *Computer Science (CS) Curricula 2013* report states, CS programs should prepare students "for the workforce in a more holistic way than simply conveying technical facts. Indeed, soft skills (such as teamwork, verbal and written communication, time management, problem solving, and flexibility) and personal attributes (such as risk tolerance, collegiality, patience, work ethic, identification of opportunity …) play a critical role in the workplace." It also states that CS programs must "expose students to multiple programming languages, tools, paradigms, and technologies as well as the fundamental underlying principles". Meeting all of these curricular goals is a challenge, especially for small CS programs, where resources are limited.**

This paper describes a model for enabling student innovation through the use of student-led projects, across the CS curriculum, within several foundational CS courses and as part of the senior design course. To illustrate how this model incorporates emerging technologies, sample student-led Virtual Reality (VR) projects are described. Results show student-led projects promote learning and help students express creativity and innovation while developing their soft skills and personal attributes. Additionally, this approach instills a culture of creativity and innovation embraced by the CS student body, where advanced students assist newer students as they embark on their journey; and it has had a significant impact on retaining CS students.

Keywords—student-led projects; project based learning; virtual reality; computer science education; innovation; retention; creativity

I. INTRODUCTION

Student innovation can be, and must be, nurtured in CS programs. The CS2013 curricula goals embodies this need and it provides several example courses and curricula to guide CS programs [1].

Yet, as [2] finds, many CS class activities are primarily understood as "solving the teacher's problems". Their research concluded that "a change towards an intentional consideration of creativity in CS classes is needed. This includes hands-on and discovery learning, assigning open tasks, as well as product-orientated tasks in formal learning settings." The world is moving from a knowledge-based economy (left-brain skills) to a creativity-based economy (more right brain skills), and practicing innovation is key to developing right brain skills [3].

Computer science education should be "recentered around a theme on innovating" [4]. "Innovation can be learned and a spirit of innovation can permeate throughout our courses, starting from the first year and building up expertise in innovation through to the final year. Starting early would help win new hearts and minds to computing" [4].

To inspire today's students, we need to ensure that we are aware of their learning behaviors and preferences [5]. Current students want hand-on exploration. They know how to use online resources. Teachers no longer need to be the information providers, but should become facilitators that inspire student innovation and creativity in an inquiry-driven project-based environment. Teachers should provide students with opportunities to learn from authentic experiences [6].

A typical software company is looking for people with unusually high levels of skills and creativity, because those are the employees one needs to create a successful product [5]. However, "computing curricula focus on producing graduates with a specific set of measurable skills. Although acquisition of skills must be part of the training of any artist, academic programs that seek to inspire creative brilliance look quite different from those that one finds in traditional engineering education. If we want to produce software artists, it might be useful to experiment with alternative educational strategies similar to those used to prepare artists, writers, and musicians. Such programs are likely to be more effective not only in producing the kind of graduates the industry seems to want but also in attracting students to the discipline" [5].

Retaining students is critical to meeting industry demand. Retention continues to be a challenging issue in computer science [7]. A great deal of research has focused on introducing CS to K-12 education, improving the CS0/CS1/CS2 level courses, or incorporating CS into general education [2, 5, 7].

Few CS programs have woven a combination of pedagogical approaches that foster innovation and creativity while being sensitive to how students learn with in-demand technology and skills throughout the curriculum [8]. The model described in this paper is a holistic approach that emphasizes creativity and innovation across the CS curriculum while increasing retention. The key approach to this model is the use of student-led projects

(SLP)s. This paper illustrates student innovation and creativity with example Virtual Reality SLPs.

II. Project-Based Learning and Student-Led Projects

One of the best ways to have students engage and innovate with the material is through the use of project-based learning (PBL). PBL can provide ancillary benefits such as improved student creativity, self-image, and motivation [9]. It has been shown to have workforce preparation, academic program retention and knowledge retention benefits. It can be incorporated into a conventional course or a PBL-style course could be developed. Students may, alternately, have a PBL experience through an independent project, as part of a senior capstone course or via activities they engage in for extracurricular educational enrichment [9].

While most CS programs use PBL in a capstone course, it is done in a variety of ways. It often involves locating real projects in the community. However, this is often too time consuming. Other capstone courses have students working individually on imaginary projects, giving them a chance to build something they own. Unfortunately, this may give students a poor understanding of the real world [10]. Many projects currently used in CS curricula lack both the "fun factor" needed to engage students, as well as the practical realism of engineering projects that include other computer science disciplines such as Software Engineering, or Human Computer Interaction [11].

Student-led projects (SLPs) are a form of PBL that offer a great degree of flexibility. SLPs often start in response to an external stimulus (such as a competition) or simply based on students' desire to learn more about a topic [9]. Students select projects that are meaningful and interesting to them. SLPs provide opportunities for students to be creative, which has been shown to increase intrinsic motivation [12,13]. Motivated students, no matter their knowledge, skills, or experiences, are more likely to succeed. Enhancing student motivation leads to better and more effective learning; it maximizes their potential [14]. Additionally, SLPs provide excellent opportunities for students to build their resume, demonstrate innovation, and investigate potential research opportunities.

Pragmatically, most departments are under pressure to reduce their course delivery costs, but techniques to motivate all students usually increase costs because they require the development of new material, running multiple level classes, and offering differentiated material [13]. SLPs provide student motivation, without the need to develop new courses.

III. A Cross-Curricular Model that Fosters Innovation

Nurturing innovation involves exposing students to the curricular concepts (what), having them engage with the material (so what), and then empower them to do something with what they have learned (now what) [15]. The approach described here follows this framework. Courses are designed to expose students to the body of knowledge as specified in the CS2013 guidelines [1] (what); provide assignments for them to engage/practice (so what), and then empower them to innovate with the material (now what).

The model proposed in this paper is based on the premise that it is best to expose, engage, and empower students within several courses, rather than to leave the empowerment phase to a single capstone course. As such, it utilizes PBL and SLPs in many ways, across the CS curriculum, embedded within several conventional courses, and as part of the senior capstone course.

This model was developed over numerous years of teaching university CS courses. In 2003, the authors started teaching the courses used in this curricular model. From 2003-2006, a traditional curriculum was used. From 2007-2010, project based learning was implemented in several of these courses. This included game and mobile app development and robotics competitions. During this timeframe, the projects were designed by the instructors. Since 2011, the proposed model which makes use of SLPs to empower students, has been used at UNK.

A summary of the model is shown in Figure 1. The model utilizes SLPs to empower students in several courses, including Data Structures & Algorithms, Artificial Intelligence, Software Engineering, and the senior capstone course. Within each course, instructors have the academic freedom to expose, engage, and empower students in their own way. Students see multiple ways to approach projects, which helps them develop flexible problem-solving skills.

This model uses an iterative process. The complexity expected and length of time students spend on their SLP in each course grows as they mature in the program. During the first year, student empowerment with SLPs within a course is limited to a 2-3 week project. During the second year, students are given more time to complete their SLPs, and the project complexity expectations deepen. By the senior capstone course, students are empowered to complete their project over the entire semester.

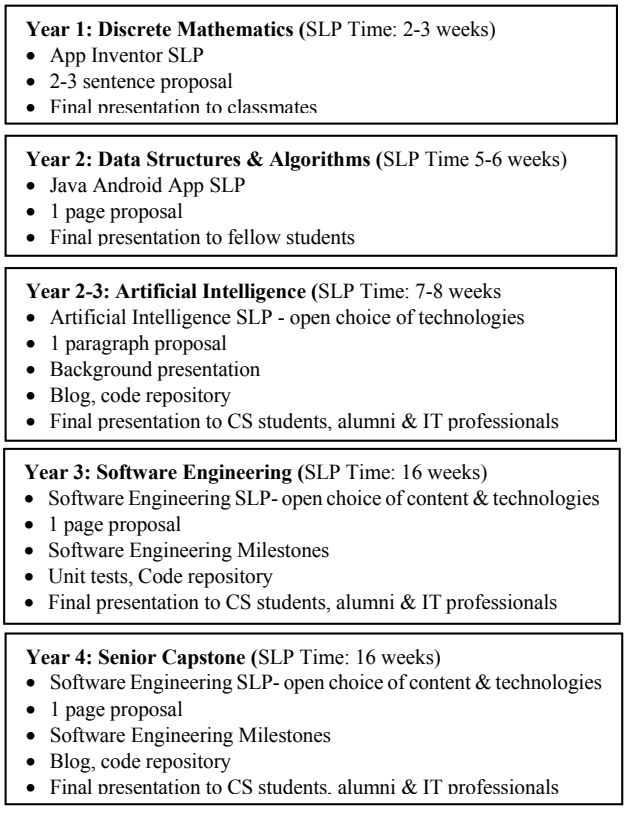

Fig. 1. Cross-Curricular Model using Student-Led Projects

The proposed model is fairly unique in the use of SLPs throughout the curriculum, but is similar to [16, 17] as it provides opportunities for implementing software engineering practices through the use of SLPs. Similarly, [18] describes changes one CS program made to encourage creative, hands-on learning by creating more laboratory space and using projects that encourage exploration across multiple courses. Their results suggest that these changes improved student skill and willingness to deal with new problems and technologies, and increased student retention.

There is always a balance between providing assignments that are focused on particular curricular topics and ones that are more open-ended [19]. The goal for instructors is to have students invested in their work. By using SLPs across the curriculum, students focus on a particular curricular topic as they are exposed to it and as they engage with the material in the completion of their project. Students gain confidence that they can be successful with learning new technologies. As [4] states, "The first challenge is to embed the foundational practices of innovation into the curriculum, so that students learn innovation by doing, without necessarily being aware they are engaged with systematic processes. The intention is that innovation should become an essential aspect of their attitude of mind. Seeking opportunities for innovation can become a way of life for students, ingrained from the very start."

In courses that use SLPs, instructors assist students early in the semester to brainstorm and develop ideas that can be implemented. Students are provided information on appropriate and inappropriate projects, as well as example projects from previous semesters. While students must complete a SLP related to the goals of the individual course, the hardware and software platforms are not specified for the upper-level courses. Students choose many different technologies, of which Virtual Reality (VR) is one area students may use. Mobile app and game development are other common choices. Students are allowed to use online resources they legally find. Student projects often use frameworks and existing online code to complete their SLPs. (This has its advantages and disadvantages as addressed later.)

For SLPs that focus on emerging technologies such as VR, it is hard to develop an accurate project management plan for successful implementation. The fear of the unknown can prohibit students from considering a project using an emerging technology, especially if easier alternatives can meet the course requirements. Additionally, inexperience with the new tools can cause students who start a project to get overwhelmed and lose the confidence and drive to complete the project.

For students to be successful, risks must be managed [20]. The key risks are estimating the time projected to complete tasks; estimating task complexity; identifying required tasks; and identifying knowledge, skill and experience gaps of students (and faculty) related to developing for an emerging technology, such as the VR platform.

To mitigate the risks students are willing to take, they are reassured that a failed project can still be a fully successful learning experience. It is only by failing to try that the student's grade is a failure as well. Instructors encourage students to try something fun, unusual and interesting – even if they only get part way done. Sample projects from previous students are provided to illustrate that "A" projects do not need to be fully functional when completed. For example, a previous student in Artificial Intelligence designed an A* Rubik's cube solver. The resulting algorithm was only successful at solving the cube's top level. This project is provided as a sample successful project, even though it was not fully functional.

The complexity of the project is a graded component. Thus, students who choose a more complex project will not be penalized when they do not get all of the project components implemented. Students are told that this is the time in their life to experiment and "play" as they learn. Similar to [12] and [18], this approach increases student willingness to tinker with something new, and establishes a culture of innovation.

At the same time, it is important to cover theoretical CS concepts and ensure that projects follow software engineering methodological principles [21]. Milestones are used to help students stay on track and to ensure that they are implementing project management and software engineering principles. Each course provides plenty of graded components for the project which help the students demonstrate their communication and other professional skills, and provides opportunities for them to manage their project and to reflect on the experience [22].

A. Discrete Math (First Year)

The proposed model starts with using SLPs in the freshman-level Discrete Mathematics course. Students typically take this course the second semester of their freshman year at UNK, at the same time that they take the CS II course. In addition to exposing students to the traditional discrete mathematics topics, students are also exposed to mobile app development using App Inventor [23]. Students also use the Project Euler problems [24] to engage with the discrete mathematics concepts.

Given the nature of the course, only a small portion of the course time is dedicated to empowering students with a SLP, and the project complexity is minimal. The final project is an app created over the last 2-3 weeks of the semester. The proposal is simply a few sentences that explains the app they are creating. During finals week, students present their projects to each other. These projects help students to start thinking creatively and independently at an early juncture in the educational career, while introducing the software engineering processes. Additionally, it sets the tone of the CS program, where innovation is encouraged, and fear of failure is minimized.

B. Data Structures & Algorithms (Second Year)

Students in the second year complete the data structures and algorithms course. In addition to engaging students with problems from the textbook [25], students are required to complete several mobile app development labs available online, created for the Cal-Poly CPE/CSC 436 Mobile Application Development course [26].

The SLP starts at week 11 of the semester and has two parts. The first is a formal project proposal (approximately one-page in length) that describes what the student intends to create for an Android application. The app must be of a similar scale to the advanced Cal-Poly labs that they have already completed.

In the proposal, students are required to include a prioritized list of features that the app will include. Students need to make

sure that what they propose is feasible given the time constraints. Students are instructed to ensure that the list has the most critical features (i.e., the features that the app absolutely must have in order to function) at the top. Each of these features should also have a date by which the student expects to have it completed, along with an explanation of data structures and basic algorithms that are planned. They are also required to include the date by which the app will be delivered. The proposal must be submitted no later than one month before end of the semester.

The second component is the completed app, which must be submitted during finals week. Students must submit both the app and the documentation (javadoc) for their programs and methods. Students present their apps to each other during the final exam time. Students are graded on their proposal, the complexity of the project as well as its appearance and correctness, and the completeness and syntactic/grammatical correctness of the documentation.

Students love the freedom to be innovative, and their app choices demonstrate their individual creativity and interests. As a second SLP, innovation is starting to become a way of life for students. Additionally, the software engineering processes and data structures and algorithm principles are reinforced.

C. Artificial Intelligence (Second or Third Year)

CS students at UNK usually complete the Artificial Intelligence (AI) course during their second or third year. Students complete several written assignments, and implement search algorithms, a robotics project, and a logic project. They are empowered to complete an independent SLP over the last half of the semester. For the SLP, students submit a brief proposal with basic AI ideas and resources they will be using. The instructor reviews these proposals to gauge if the project is doable within the timeframe and if the project meets the overall goals. The proposal may need to be resubmitted numerous times before it is approved. The projects must have core features that demonstrate an understanding and implementation of AI while leaving room for individual creativity. Sample projects from previous semesters are provided to help students determine the complexity requirements and to spark innovation.

Students give a background presentation to the class part-way through the semester. Students introduce the goals of their project and describe the AI ideas being implemented. Because of the wide variety of available online AI resources, students often make use of these external resources. During the background presentations, students describe the resources they are using in their project. Students give each other feedback and suggestions during these presentations.

Students keep a weekly blog to report progress. These provide opportunities for the instructor and other classmates to provide feedback and suggestions. One of the most important reasons that professional software engineering project fail is poor reporting of the project's status [27]. Because of the poor reporting, the management does not know the state of the project and thus does not execute the right actions. Student software development projects can fail for same reason, thus blogging is a key component to the success of SLPs. Blogging also provides students with a place for reflection and analysis. Students are required to submit at least one entry a week for the last five weeks of the semester, along with a final blog post during finals week. Blog posts should be well written and explain the project and the on-going progress. They provide the background for the project and links or references used. Screen shots or pictures are used to enhance the blog postings. Students are expected to explain the algorithms and data structures used, along with the experiments conducted. They are also expected to explain lessons learned, discuss future enhancements that should be made, and future testing that should be performed.

A key risk mitigation factor is ensuring that project knowledge is well documented [9]. In our model, code repositories are used. Together with the blog and internal code documentation, the project knowledge is well documented and accessible.

At the end of the semester, students complete a code-walk through with the instructor, a final project presentation, along with their final reflection blog post that also includes an executable download, a link to a video demonstrating the project, or some way for the viewer to see the project in action. (Rubrics for these outcomes are available upon request.)

The final project presentations are given in a professional conference style. Business representatives, alumni, and the entire CS student body are invited to attend. This provides an opportunity for students to disseminate results and gain recognition.

By now, innovation is engrained as a norm for students at this stage in their CS program. Even the presentations do not seem to cause difficulties to the students. Additionally, students are gaining an understanding of the software engineering process while utilizing external resources, and gaining an understanding of project and time management, while implementing AI concepts.

D. Software Engineering (Third Year)

CS students at UNK also usually complete Software Engineering during their third year. Student engagement and empowerment occur through a semester long SLP. The project proposal is submitted within the first two weeks of the semester. It requires students to describe what they intend to create for their semester project. As with the Data Structures & Algorithms project, students are expected to pretend the instructor is a potential customer that the student is trying to impress with his/her awesome idea. Students are instructed to provide a list of features, with the most critical features (i.e., the features that the absolutely must have in order to function) at the top. Each of these features should also have a date by which the student expects to have it completed, along with an explanation of the basic algorithms that are planned. They are also required to include the date by which they will have their project complete. As with the AI project, sample projects from previous semesters are provided to help students determine the complexity requirements and to spark innovation.

Other software engineering milestones that students must meet include: a submission of unit tests for several non-trivial project methods; a detailed timeline for the subtasks of the project; and project updates throughout the semester. At the end of the semester, students complete a code-walk through with the instructor, a professional final project presentation, and submit

both their project and its documentation. Occasionally, students will tie their AI and Software Engineering projects together, such as an Android app called "Inflatable Defense" created in spring 2013, and available on the Google Play Store [28].

E. Senior Capstone Course (Fourth Year)

As a small department, students in both the Information Technology (IT) and CS programs complete the senior capstone course together at UNK. The goals of this class are to improve research, communication, team development and project development skills. Other goals are to learn professional ethics and to reinforce prior CS/IT knowledge by creating a capstone project. Students complete written assessments for the ethical concepts that measure written communication skills as an outcome of the CS/IT program. They also work in teams to present and lead course discussions on the ethical topics.

Students usually complete their SLPs in teams. Because of the size of the department, students are allowed to work independently, if they are interested in a certain technology, and no other student has a shared interest. However, the ethical components of this course are still completed in teams.

The capstone project follows a similar framework as the AI SLP, but students have the entire semester to complete the project. Students submit proposals for the instructor to review; complete the project over the course of the semester; and keep a project blog. Each team provides a weekly update to the instructor. Project milestones must be met throughout the semester, and a project schedule is maintained.

The blog serves as a record of the project activities. It starts with a one-page summary of the project, which includes the title of the project, the team members, and a 1-2 paragraph description of the project. Students make weekly entries that include what was accomplished, how much time was spent, and anything else the student would like to add. Statements about what they would do differently next time are encouraged. A list of references along with a short entry as to the content in the reference must be included on the blog. The instructor checks the blog at least twice throughout the semester. The blogs are kept confidential unless the student gives permission to share information. Blog grades are based on the quantity and quality of detailed information that is recorded.

Each team makes a formal, public presentation of their project during finals week. This is used an assessment tool to measure students' oral communication as an outcome of the program. All team members are to participate in some way. As with the AI project presentations, business representatives, alumni, and the entire CS student body are invited to attend.

At the end of the semester, students complete a code-walk through with the instructor, and submit the final project with its documentation along with their final reflection blog post that also includes an executable download, a link to a video demonstrating the project, or some way for the viewer to see the project in action. The project is used as an assessment tool to measure student's ability to meet the assessed goals of the CS/IT programs.

As the fifth SLP that CS students have completed in their CS major, the student innovation and creativity in the senior capstone projects is outstanding. A sampling of student projects are available online [29]. Using SLPs not only enables innovation, it also enables the students to lead the way in introducing new technologies.

IV. Example Virtual Reality Student-Led Projects

There is a huge amount of development interest in VR across multiple industries, including video streaming, gaming and simulated learning. Even though PC, web and mobile are the top platforms for software development, it is important for CS programs to expose students to VR as a development platform.

VR replicates an environment that simulates a physical presence in the real world or an imagined world, allowing the user to interact in that world. VR artificially creates sensory experiences, which can include sight, touch, hearing, and smell. The simulated environment can be similar to the real world in order to create a lifelike experience, such as in simulations for pilot or combat training, or it can differ significantly from reality, such as in VR games. Most advances are being done in the entertainment industry, but VR will be important to many fields, especially in education, training and simulation, tourism, patient care, mental therapy, architecture and modeling [30].

Developer support for Oculus Rift [31] and other virtual reality headsets is at an all-time high [32]. Oculus' momentum clearly demonstrates technology has reached a point, in terms of quality and price, where these devices can be made available to the public in an attractive format. And, more important, they demonstrate new possibilities that can spark imagination [33]. Developers are also pretty keen on VR being a long-term platform, with 75 percent stating it as a long-term sustainable business. Some analysts estimate the VR industry will be worth $62 billion by 2025 [33].

Using VR in SLPs is one way to introduce it to CS students, as it is challenging to fit VR into the CS curricula. Using VR as a development platform requires the use of innovation, creativity, and problem-solving skills.

Starting in 2014, the UNK CSIT department adopted VR resources and lab space for student projects. With these resources, CSIT students quickly became innovators in the VR world. Each new version of VR hardware and/or software and each new semester brought opportunities for students to try new things and express their creativity. Additionally, each student project inspired more students to try VR development.

Through this entire process, the faculty have fostered student innovation, but have not been the information providers – students have relied on online resources and each other for information. The faculty have provided students with opportunities to learn from authentic experiences, and allowed students to explore what is relevant to them.

The first CSIT VR project was a snowboarding simulation completed in spring 2014 as Software Engineering SLP. It used the first generation Oculus Rift VR display for head tracking and display along with the Wii Balance Board [34], to control the in-game snowboard. It was developed with the Unity Engine [35], and used Blender [36] for 3D modeling. An image of a student using the snowboarding simulation is shown in Figure 2, and a video demonstration is available on YouTube [37].

In the spring of 2015, two students created VR projects in the Artificial Intelligence (AI) course, using the Oculus Rift Development Kit (DK) 2. One was a VR first person shooter game "robots versus humans" [38] for the Oculus Rift KD2. A screen shot shown in Figure 3. It used the Unity Game Development Engine, along with TF3DM [39] and Turbo Squid for 3D modeling [40]. The most challenging aspect of the development process was making the robot have enough intelligence to target the human controlled player, but not enough intelligence that the human controlled character didn't have a chance. The other VR SLP was a VR motorcycle racing game also created for the Oculus Rift DK2. It involved a player surviving on a motorcycle while avoiding walls and an AI motorcyclist. A video demonstration and explanation of the game is available on YouTube [41].

The fall 2015 semester brought about more student-led VR projects. The student who created the motorcycle racing game in AI, set out to improve it in the Computer Graphics course. Unfortunately, innovating in the VR world has its challenges. As the game was ported to new Unity 5 engine, the motorcycle which worked perfectly in the previous version, now moved straight up instead of straight forward. This problem was quickly fixed. The ability for the human player to jump over walls was added and other adjustments to the environment were made. Allowing a SLP to continue multiple semesters encourages students to start a cutting edge VR project, as they know it is worth the initial effort and time investment, and it allows them to get deeper into the technology and practice the process of iterative development.

In fall 2015, a team of two students created a VR project in their Senior Capstone course. This was a VR 3D boxing game for the Oculus Rift. It was built using Unity with Leap Motion [42], using Blender for the 3D modeling. The Leap Motion allows the VR system to "see" the player's hands. Unfortunately, it only recognizes a hand when all five fingers were visible. This made it difficult to see the player's fist, as needed for their boxing game. Their game turned into more of a "slapping" game. One of the students is shown in Figure 4 presenting their game at the Fall 2015 Project Demonstration Day. The Oculus Touch [43], which is a handheld tracked controller that brings the player's hands fully into VR, is planned to be released in the fall 2016. (This technology would have allowed the boxing game to function more realistically.)

Fig. 2. Snowboarding Virtual Reality Student Led Project

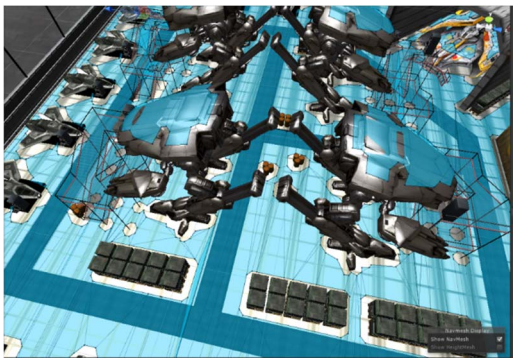

Fig. 3. Screenshot of Humans vs. Robots VR Project

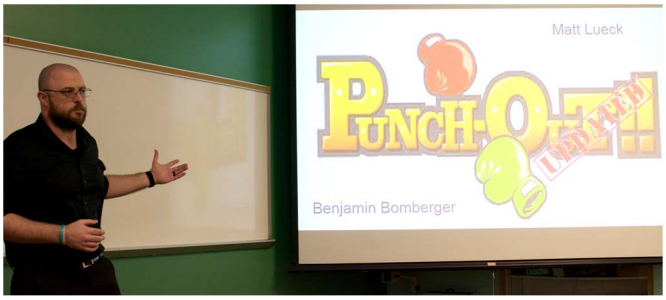

Fig. 4. Sample Student Presentation of VR Student-Led Project

V. RESULTS

A traditional CS curriculum was used at UNK from 2003-2006. During this time frame, of the CS students who completed the Discrete Math course, only 54% were retained to graduation. When PBL was implemented in several of these courses from 2007-2010, the 6-year graduation rate improved to an average of 64%. Since 2011, the proposed model has been used. Because a majority of these students are still in the program, the persistence rate is used for this time frame as it reflects students who are currently in the program or graduated from the program. The persistence rate has averaged over 80% during this time frame. Yet, the most recent average 6-year graduation rate for UNK is 57% and the most recent 2-year persistence rate at UNK is 70% [44], while the average 6-year graduation rate for public institutions is 58%, and for computer science/information systems majors in general is 59% [45].

Since most attrition occurs between the first and second years [7], we compare this persistence rate to the earlier retention to graduation rates. The 2011-2015 persistence rate is a significant improvement in retention (using the Chi Square statistics at a p-value of .10) over the earlier retention rates. Retention rates and the retention trend line are shown in Figure 5. As shown, the retention rates varied year by year, but generally increased. The three different time frames (2003-2006; 2007-2010; and 2011-2015) are shown in different gray-scale colors to visually show their groupings. The trend line shows increasing retention rates as well.

Some student comments from the last five years in the Discrete Math course include: "I enjoyed some of the projects that we did because they were interesting and let me try new things." "Allowing students to solve problems in whichever they choose. I loved this course and professor!" "Dr. Hastings provided a learning environment that was fun to attend. I enjoyed doing the Project Euler problems and Android app

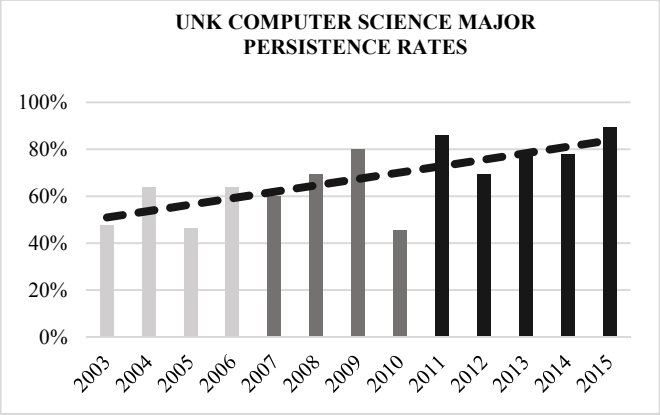

Fig. 5. UNK Computer Science Retention Rates 2003-2015

development. I enjoyed that Dr. Hastings did not require a specific language for the Project Euler problems which opened the doors to learn and expand my abilities in different languages."

Students in the Data Structures and Algorithms course have commented: Enjoyed having last month of class to work on final app." "Assignments were nicely laid out and effective." "Learning things that are relevant to the tech world, such as mobile dev & version control."

Students in the Artificial Intelligence class commented, "Allowing students to choose their projects allowed passion to develop which as extremely useful."" I learn the best by doing things. I'm glad this class implemented major projects instead of tests to reinforce knowledge." "I like the inclusion of the competition and final project."

Software Engineering students commented "Being able to pick a project that interested us was great." "I believe doing a software engineering project ourselves was a good idea."

Senior Capstone students commented, " I loved getting to pick a project of interest to me and having a semester to work on it." "Picking a project is a great choice for students. Learned a lot." "Enjoyed have the freedom to choose my own project." "Cool Beans! This was fun!"

VI. Discussion

Using SLPs within the context of a course, all students are empowered the opportunity to autonomously complete their own projects. Additionally, when lab time is given to work on these projects, the instructor is available to answer questions as they arise. This also helps the instructor make sure that the projects are on track to be completed. Students also tend to help one another as they learn something new.

Reference [12] found, "it was most rewarding for students to strive for good working software, based on self-chosen and meaningful tasks, even if the resulting products did not have any outside value." This research have found similar results.

Reference [5] stated that to inspire today's students, "we need to ensure that we are aware of our new generation of students learning behaviors' and preferences." This research found that using SLPs fits well with today's students. They want hand-on exploration. They know how to use online resources. This research found that students are enthusiastic and motivated when they lead the project, from scope definition to implementation.

Many CS programs used Special Topics courses to introduce students to new concepts and technologies or specific courses focused on introducing creativity and innovation to students [46, 47]. As a small CS program, using SLPs allows students to work with new technologies, without having a specialized course on a topic. Additionally, using SLPs does not require changes to curriculum, which is often a slow process.

Using SLPs has instilled a culture of innovation and creativity embraced by the CS student body. As new technologies are introduced, CS faculty and students learn together. SLPs provide a curricular shift from a teacher-centered focus to a student-centered focus which helps students develop as life-long learners.

The proposed model uses SLPs in five key CS courses. While the courses selected worked best for the CS program at UNK, other programs may find that different courses would work better for them. However, the model works because SLPs are used in multiple courses, over the course of the entire CS program, with increasing innovation and creativity requirements.

There are some major advantages that SLPs have over student research. Student research requires one-to-one supervision that takes up a significant amount of academic time. Also, research students need to be paid some sort of living expenses either at an hourly rate, by an honorarium or scholarship. For these reasons, most departments offering such experiences do so only to selected students [13].

As mentioned earlier, retention continues to be a challenging issue in computer science programs. The proposed model of cross curricular use of SLPs was shown to increase the persistence rate to 80% for CS students at UNK, even when the most recent 2-year persistence rate at UNK was 70%. The authors recognize that some attrition is still likely to occur for the group of students who took the freshman-level Discrete Math course between 2011 and 2015, and the 6-year graduation rate will likely be lower than the current 80% persistence rate. Some attrition occurs as students enter the IT profession before graduation. For example, in each of the 2011 and the 2012 classes, there is a student who gained full-time employment as an IT professional while just shy of graduation. The authors are unsure if these students will choose to graduate. In fact, because of the shortage of IT professionals and the high salary rates, it is not uncommon for this to occur.

A. Outreach & Entrepreneurial Opportunities

There are many advantages and benefits to empowering students through SLPs. Similar to undergraduate research, academic departments benefit from the reflected "glory" of successful projects which can be reported on web sites and in news articles. Student successes in SLPs help encourage and motivate other students as well.

For example, the snowboarding VR project was presented to local businesses and CSIT students at the CSIT Spring 2014

Project Demonstration Day; and at outreach events for middle and high school students at the 2014 Nebraska Broadband Conference and at the 2015 First Lego League Robotics Competition in Kearney. The motorcycle racing game and the robots versus humans game were demonstrated at a conference for high school students in spring 2016 [48].

The student who created the motorcycle racing game is shown in Figure 6 pitching his idea for a VR game development company. He won the 5th annual Central Nebraska business idea contest in October 2015 [49]. Notice that he is the same student trying out the VR project three semesters earlier in Figure 1. This illustrates how the students who complete VR SLPs inspire and teach the younger students to also try VR development.

At UNK, using SLPs has helped students be successful in their careers. In the past several years, 100% of all dedicated CSIT graduates have gained professional employment in their major area of study. Additionally, in the past several years, 100% of the sophomore-level or above CSIT students (who were U.S. citizens) seeking internships have been successful. This includes full-time summer internships, or part-time internship/IT-related employment in the area throughout the academic year. Businesses who have hired at least one CSIT graduate in the past five years, return to CSIT when seeking to fill new positions. Businesses are also started to provide endowed scholarships, including a $300,000 endowment from Buckle Inc., a clothing company headquartered in Kearney, NE.

B. Lessons Learned & Future Work

Managing independent SLPs is challenging. It can be difficult to assess the student's progress, particularly if it takes a while for him or her to get the project up and running [17]. The projects need to not be too shallow and yet not be too idealistic either. Additionally, there is a limit to the number of SLPs that an instructor can manage. At UNK, the authors found that using SLPs with a class of more than 30 students was hard to manage. Also, SLPs often require unique hardware/software purchases, which can be costly. Resource limitations need to be discussed before students start a SLP.

By using SLPs in multiple courses, students get better at developing a manageable project scope. Additionally, students who find out that they were too idealistic in their expectations can still "complete" a project, and yet have a great deal of the initial project unfinished.

Instructors also get better at managing SLP with more experience. Using multiple graded components, blogs, and expressed milestones are key elements to helping manage successful SLPs.

Fig. 6. Student Pitching VR Game Development as a Business

Another issue faced when using SLPs comes with allowing students to use frameworks and existing code available online to complete their project. The instructors have had a few students who found code online, and simply wanted to call that their project. Using milestones with expectations of code walkthroughs and blog posts that explain weekly progress address these issues. By using SLPs, there is little opportunity for students to copy each other's work.

At times, students were overwhelmed with the material they found online, and struggled to figure out how to use it. Software developers can attest as to how difficult it is to read someone else's code, especially if it is poorly documented. Code updates and blog posts help identify these issues. Plus, as [6] stated, "Discovery learning [such as through SLPs] shifts an instructor's task away from preparing and reciting lectures, and towards free-form interaction with students." Thus, it is important for instructors to maintain the interaction with the students to make sure each student is on track to be successful.

Thus far, the use of SLPs in the CS programs at UNK has been limited to on-campus courses. However, the UNK Database Systems course, which utilizes a service-learning project, is offered both online and on campus. Virtual meetings are required to maintain interaction with the individual online students. Coordinating and conducting these meetings takes additional time commitments from the instructor.

Future work should explore ways to expand this model to online programs, or eventually to massively open online courses. Future work should also explore ways to expand this model to other technical programs, such as computer engineering or information technology programs.

VII. CONCLUSION

Empowering students through the use of SLPs across the curriculum embodies the goals of the CS2013 curricula guideline. It is a holistic approach that goes beyond exposing students to technological facts. It helps students develop their verbal and written communication, time management, problem-solving skills. It helps students develop risk tolerance, innovation, and creativity.

The proposed model follows a framework where students are exposed to new CS concepts; provided opportunities to engage with the material, and then empowered through SLPs to innovate. The persistence rate increased from 54% to 80% when this model was implemented. By iteratively using SLPs across the curriculum, students gain confidence that they can be successful with learning new technologies. Students are enthusiastic and motivated when they lead the project, from scope definition to implementation. It allows students to work with new technologies, without requiring curricular changes or specialized courses. Additionally, it provides for outreach and entrepreneurial opportunities.

REFERENCES

[1] Computer Science Curricula 2013, retrieved February 16, 2016 from http://www.acm.org/education/CS2013-final-report.pdf.

[2] M. Knobelsdorf and R. Romeike. 2008. Creativity as a pathway to computer science. SIGCSE Bull. 40, 3 (June 2008), 286-290.

[3] A. L. Tharp. 2007. Innovating: the importance of right brain skills for computer science graduates. In Proc. 12th SIGCSE conference on Innovation and Technology in Computer Science Education (ITiCSE '07). ACM, New York, NY, USA, 126-130.

[4] P. J. Denning and A. McGettrick. 2005. Recentering computer science. Commun. ACM 48, 11 (November 2005), 15-19.

[5] D. D. Garcia, R. Cutler, Z. Dodds, E. Roberts, and A. Young. 2009. Rediscovering the passion, beauty, joy, and awe: making computing fun again, continued. SIGCSE Bull. 41, 1 (March 2009), 65-66.

[6] D. Baldwin. 1996. Discovery learning in computer science. In Proc. 27th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '96), Karl J. Klee (Ed.). ACM, New York, NY, USA, 222-226.

[7] T. DeClue, J. Kimball, B. Lu, and J. Cain. 2011. Five focused strategies for increasing retention in computer science 1. J. Comput. Sci. Coll. 26, 5 (May 2011), 252-258.

[8] S. Azadegan, J. Dehlinger, and S. Kaza. 2014. Incorporating mobile computing into the CS curriculum. In Proc. 45th Technical Symposium on Computer Science Education (SIGCSE '14). ACM, New York, NY, USA, 735-735.

[9] J. Straub. 2015. Involvement of Undergraduate Students in Research: A Comparison of Course Research Components, Paid Research Activities, Student-Led Projects and Independent / Directed Study Courses, In Proc. of Midwest Instructional Computing Symposium, 2015, micsymposium.org/mics2015/ProceedingsMICS_2015.

[10] K. Wright. 2010. Capstone programming courses considered harmful. Commun. ACM 53, 4 (April 2010), 124-127.

[11] K. Claypool and M. Claypool. 2005. Teaching software engineering through game design. SIGCSE Bull. 37, 3 (June 2005), 123-127.

[12] R. Romeike. 2008. Towards students' motivation and interest: teaching tips for applying creativity. In Proc. 8th Intl Conference on Computing Education Research (Koli '08). ACM, New York, NY, USA, 113-114.

[13] J. Carter, D. Bouvier, R. Cardell-Oliver, et al., 2011. Motivating all our students?. In Proc. 16th Conference on Innovation and Technology in Computer Science Education - working group (ITiCSE-WGR '11), Adams and Jurgens (Eds.). ACM, New York, NY, USA, 1-18.

[14] A. Carbone, J. Hurst, I. Mitchell, and D. Gunstone. 2009. An exploration of internal factors influencing student learning of programming. In Proc. 11th Australasian Conference on Computing Education - Volume 95 (ACE '09), Hamilton and Clear (Eds.), Vol. 95. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 25-34.

[15] S. Sakai-Miller. 2016. Innovation Age Learning. International Society for Technology in Education.

[16] L. B. Sherrell and S. G. Shiva. 2005. Will earlier projects plus a disciplined process enforce SE principles throughout the CS curriculum?. In Proc. 27th Intl Conference on Software Engineering (ICSE '05). ACM, New York, NY, USA, 619-620.

[17] M. R. Hribar. 2005. Sure Fire Programming: a general framework for independent projects in Computer Science. J. Comput. Sci. Coll. 21, 1 (October 2005), 257-266.

[18] G. Lewandowski, E. Johnson, and M. Goldweber. 2005. Fostering a creative interest in computer science. In Proc. 36th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '05). ACM, New York, NY, USA, 535-539.

[19] F. Martin. 2006. Toy projects considered harmful. Commun. ACM 49, 7 (July 2006), 113-116.

[20] B. Boehm and D. Port. 2001. Educating software engineering students to manage risk. In Proc. of the 23rd Intl Conference on Software Engineering (ICSE '01). IEEE Computer Society, Washington, DC, USA, 591-600.

[21] A. Bobkowska. 2015. Balance Between Creativity and Methodology in Software Projects. In Proc. of the Multimedia, Interaction, Design and Innovation (MIDI '15). ACM, New York, NY, USA.

[22] C.J. Lesko, Jr.. 2009. Building a framework for the senior capstone experience in an information computer technology program. In Proc 10th ACM Conference on SIG-Information Technology Education (SIGITE '09). ACM, New York, NY, USA, 245-251.

[23] MIT App Inventor, retrieved February 16, 2016 from http://appinventor.mit.edu/explore/about-us.html.

[24] Project Euler problems, retrieved February 11, 2016 from projecteuler.net.

[25] T. Cormen, C. Leiserson, R. L. Rivest, C. Stein. 2009. Introduction to Algorithms, 3/e. McGraw Hill.

[26] D. Janzen, K. Owen, J. Reed. 2013. California Polytechnic State University CPE/CSC 436 Mobile Application Development course lab materials, retrieved February 15, 2016 from sites.google.com/site/androidappcoursev3/.

[27] P. Mäkiaho, T. Poranen, and A. Seppi. 2015. Software metrics in students' software development projects. In Proc. 16th Intl Conference on Computer Systems and Technologies (CompSysTech '15), Boris Rachev and Angel Smrikarov (Eds.). ACM, New York, NY, USA, 75-82.

[28] Inflatable Defense Android game, retrieved April 1, 2016 from play.google.com/store/apps/details?id=com.houledm.inflatabledefense.

[29] University of Nebraska at Kearney Computer Science & Information Technology Department Student Projects, retrieved February 15, 2016 from unk.edu/academics/csit/student_info/projects_research.php.

[30] Five Amazing Non-Gaming Ways People Are Using Oculus Rift, retrieved February 11, 2016 from makeuseof.com/tag/5-amazing-non-gaming-ways-people-using-oculus-rift/.

[31] Oculus Rift, retrieved March 8, 2016 from oculus.com.

[32] Game Developer Conference News and Information Blog. 2015. retrieved February 11, 2016 from gdconf.com/news/gdc_state_of_the_industry_most.html.

[33] D. M. Plasencia. 2015. One step beyond virtual reality: connecting past and future developments. XRDS 22, 1 (November 2015), 18-23.

[34] Wii Balance Board, retrieved March 8, 2016 from www.nintendo.com/consumer/downloads/wiiBalanceBoard.pdf.

[35] Unity Engine, retrieved March 8, 2016 from unity3d.com.

[36] Blender 3D Modeling software, retrieved March 8, 2016 blender.org.

[37] D. Russell. 2014. Slalom VR Spring 2014 Project, retrieved March 8, 2016 youtube.com/watch?v=KjTtZWYA04g.

[38] I. Lim. 2016. Robots Vs. Humans blog, retrieved March 8, 2016 from robotsvshumansposts.tumblr.com.

[39] TF3DM 3D Modeling software, retrieved March 8, 2016 from tf3dm.com.

[40] TurboSquid 3D Modeling software, retrieved March 8, 2016 from turbosquid.com.

[41] S. Middleton. 2015. Central NE Business Idea Contest - VR Video Game, retrieved March 8, 2016 from youtube.com/watch?v=0M_B7mCIMy4.

[42] Leap Motion, retrieved March 8, 2016 from leapmotion.com.

[43] Oculus Touch hands-on: Worth the wait? 2016. , retrieved March 8, 2016 from technobuffalo.com/2016/01/11/oculus-touch-hands-on-worth-the-wait/.

[44] University of Nebraska at Kearney (UNK) Factbook. 2016. retrieved March 8, 2016 from http://www.unk.edu/factbook.

[45] Institutional Retention and Graduation Rates for Undergraduate Students. 2016. National Center for Education Statistics, retrieved March 8, 2016 from nces.ed.gov/programs/coe/indicator_cva.asp.

[46] A. Salgian, T. M. Nakra, C. Ault, and Y. Wang. 2013. Teaching creativity in computer science. In Proc. 44th Technical Symposium on Computer Science Education (SIGCSE '13).

[47] M. Kwasnik. 2014. Nature of Creativity in Computer Science Education. Designing Innovative Workshops for CS Students. In Proc. 2014 Intl Conference on Multimedia, Interaction, Design and Innovation (MIDI '14). ACM, New York, NY, USA, Article 8, 7 pages.

[48] Career Pathways Institute Info Tech Expo. 2016. retrieved March 8, 2016 from itx16.gips.org.

[49] T. Gottula. 2015. UNK's Middleton wins business idea contest with virtual reality game, UNK News, retrieved March 8, 2016 from unknews.unk.edu/2015/11/24/unks-middleton-wins-business-idea-contest-with-virtual-reality-game.