SCALING TO MEET THE ONLINE DEMAND IN SOFTWARE ENGINEERING

Kevin A. Gary,[1,*] **Ruben Acuna,**[1] **Alexandra Mehlhase,**[1] **Robert Heinrichs,**[1] **& Sohum Sohoni**[2]

[1] School of Computing, Informatics, & Decision Systems Engineering, The Ira A. Fulton Schools of Engineering, Arizona State University, Tempe, Arizona, USA

[2] Department of Electrical and Computer Engineering, Milwaukee School of Engineering, Milwaukee, Wisconsin, USA

*Address all correspondence to: Kevin A. Gary, School of Computing, Informatics, & Decision Systems Engineering, The Ira A. Fulton Schools of Engineering, Arizona State University, Tempe, AZ 85281, USA, E-mail: kgary@asu.edu

Arizona State University's bachelor of science in software engineering is the first Accreditation Board for Engineering and Technology (ABET) accredited software engineering program offered in an online modality. ASU's online software engineering program has experienced rapid growth, to over 1000 students in a 5-year span. The program's design is the same as the on-campus offering, featuring a unique curriculum centered on a professional spine comprised of team-oriented project-based learning courses. The scale of the program and its growth, combined with a hands-on applied learning approach, creates challenges that have mandated innovative and adaptable processes to be successful. Specifically, the faculty have led a three-year effort on pedagogical innovations and internal quality process improvements to address unique aspects of online software engineering education delivery. In this paper we will present the evolution of the online program and the innovations required to support scale and growth while producing industry-ready software engineers. These innovations have resulted in an upward trend in student satisfaction, reversing a prior three-year downward trend from the inception of the online program.

KEY WORDS: software engineering education, online, education technology, software tools, pedagogy, project-centered learning, experiential learning

1. INTRODUCTION

Online engineering education presents unique challenges. Today's engineer must design robust solutions to complex problems in team-oriented environments. While software engineering does not have as many hardware requirements as most other engineering majors, it does have unique challenges due to a heavy reliance on software tools, popularity of agile methods, lack of tangible (tactile) artifacts, and emphasis on time-to-market. Further, software engineering is a popular career track due to the prevalence of software in society and a positive job outlook. Innovations in the BSSE at Arizona State University have directly addressed the challenges of scaling online software engineering education in the face of a popular and rapidly expanding program through the innovative application of technology and a focus on quality process improvement.

Software engineering is a unique professional engineering discipline in a number of ways. The primary artifacts are not tangible (Brooks and Bullet, 1987), and the overriding pressure on software businesses is usually time-to-market. Software engineering projects have historically been known to suffer from budget and time overruns, and as software plays an increasingly important role in the technologies society relies on every day, they are also susceptible to high-profile catastrophic system and security failures. The fast-paced, ever-shifting software technology landscape creates unique engineering constraints which also filter down to software engineering educators as they train the next generation of engineers.

Software engineering is a growth discipline. The U.S. Bureau of Labor and Statistics outlook reports software development as a high-growth occupation (25.6%, 2008–18), with the third highest median annual wage ($103K) (BLS, 2018). Media reports (e.g., US News and World Report, 2020) regularly cite it among the most rewarding and well-paid professions. Software engineering is seen as a fast-paced and emerging discipline (by historical comparison to traditional engineering). In the past two decades, software engineering education has grown to an established community with exposure at major conferences such as SIGCSE, ICSE, CSEET, FSE, FIE, and ASEE. The late 1990s and early 2000s saw a particular flurry of activity culminating in the ACM/IEEE recommendations for software engineering curriculum (ACM & IEEE, 2004, 2015) and the Software Engineering Body of Knowledge for professionals (Bourque and Fairley, 2014).

While software engineering (SE) is seen as one of several research subdisciplines under the umbrella of computing sciences, undergraduate degree programs are evaluated under the Engineering Accreditation Commission (EAC), not the Computing Accreditation Commission (CAC), by the Accreditation Board for Engineering and Technology (ABET). The intersection of engineering and computing foundations has several practical implications for software engineering degree programs. SE programs are typically far more

applied and industry-focused than most other subdisciplines in computing sciences and require an engineering approach with an emphasis on a measurable design process. Most students in SE programs in the U.S. complete a terminal degree program designed for graduates to enter the profession instead of continuing to graduate school. This industry focus, a strong career market, and the practical advantages of completing an online program with minimal external hardware needs other than a general purpose personal computing environment make software engineering a fast-growing online degree space.

Arizona State University's bachelor of science in software engineering is the first ABET-accredited software engineering program offered in an online modality. ASU's online software engineering program, started in 2014, has grown from zero to 1,076 students in a 5-year span (Fig. 1) while keeping roughly the same number of faculty. The program features a unique curriculum centered on a professional spine comprised of team-oriented project-based learning courses. The scale of the program and its growth, combined with a hands-on applied learning approach, creates challenges in delivery that have required innovative and adaptable processes to be successful. Specifically, the faculty have led a three-year effort on pedagogical innovations and internal quality process improvements to address unique aspects of online software engineering education delivery. In this paper we present the evolution of the online program and the innovations required to support scale and growth while producing industry-ready software engineers. This history will be

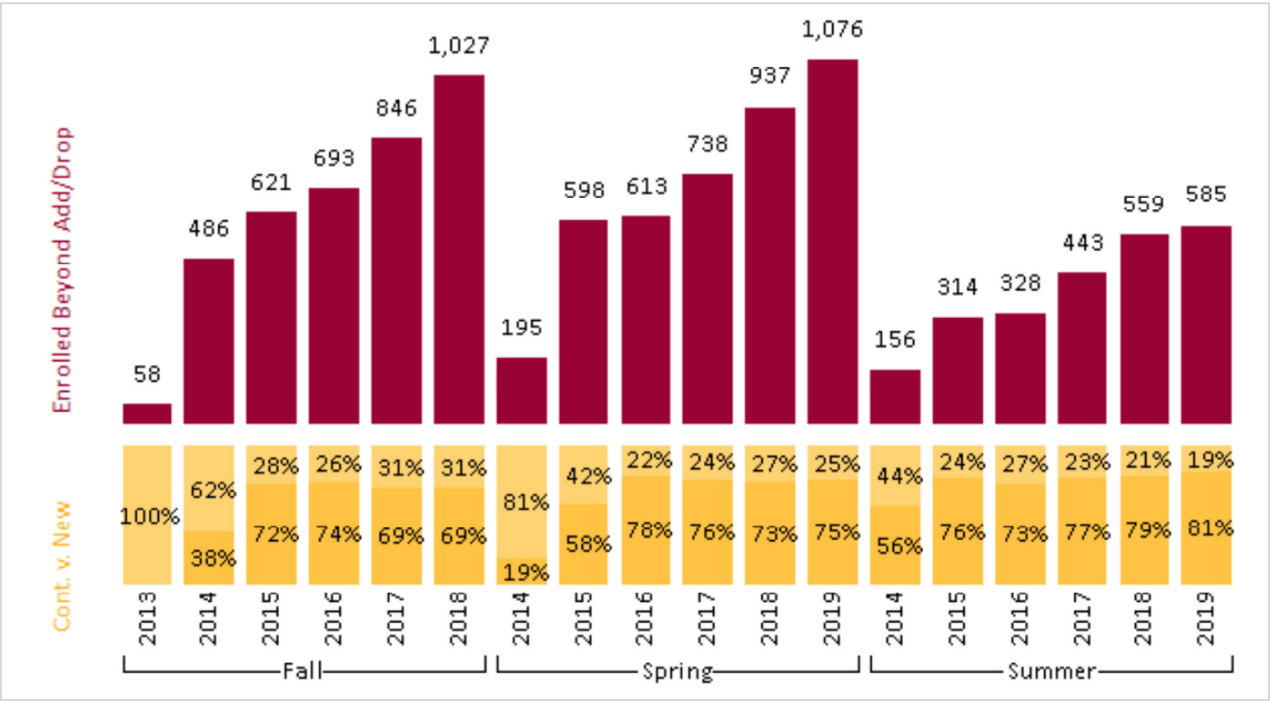

FIG. 1: The rate of enrollment growth in ASU's BSSE online degree program. The program started in 2014, the 2013–14 academic year numbers reflect recruiting numbers.

supported by analysis drawn from formal and informal student feedback instruments over time. We conclude with ideas on how to be successful going forward, and lessons for all engineering programs that may soon venture into the online space.

2. ONLINE ENGINEERING PROGRAMS

More and more students are taking online classes worldwide, either in the form of massive open online courses (MOOCs) or as part of regular coursework at a higher-education institution. The number of students enrolled in online courses at public institutions in the US has been rising annually, and it is clear that online education is now part of the mainstream educational experience in the United States. Data released by the U.S. Department of Education showed that in 2016, online enrollments dropped in for-profit organizations and gained in nonprofit institutions like Western Governors University and Arizona State University (Lederman, 2018). According to the report, about 30% of undergraduate students and 36% of graduate students in the US were enrolled in at least one online course in 2016, and at least two-thirds of these students were enrolled in public institutions of higher education (i.e., not at for-profit institutions). The number of students enrolled in online courses at public institutions in the US has been rising annually, and it is clear that online education is now part of the mainstream educational experience in the United States.

According to data on the US News and World report website (2019), as of September 2019, there are 367 US institutions that offer undergraduate programs online, of which 299 offer programs that are 100% online. Of the 367 institutions, 31 offer some stream of engineering degrees online, 21 of which are 100% online; 13 institutions offer computer science bachelor's degrees online (12 of which are 100% online); and 2 offer software engineering bachelor's degrees, both offering them 100% online.

Although a body of knowledge exists on teaching and learning online, researchers (Lack, 2013; Means et al., 2009; Wu, 2015; Beetham and Sharpe, 2020) have identified a lack of dependable research literature on online education from the perspective of pedagogy and the scholarship of teaching and learning.

Different aspects of online learning need to be explored in more detail within specific settings (Means et al., 2014, p. 35), and there is a clear need for moving away from broad (often promotional) claims about the benefits of online education and focusing on rigorous research to identify and study the aspects of online education that provide advantages to students. Through this paper, the authors hope to provide such details related to pedagogical innovations in the context of online learning and their benefits to students.

3. BACKGROUND: ASU'S SOFTWARE ENGINEERING DEGREE PROGRAM

The bachelor and master of science in software engineering (BSSE/MSSE) degrees at Arizona State University (ASU) were first offered in 2010. The degree programs were originally conceived and implemented at ASU's Polytechnic campus in recognition of the industry-focused career track typical of SE degree programs. Up to this time, software engineering and computer science education efforts focused on content taxonomies and bodies of knowledge (BOKs) (Bourque and Fairley, 2014; ACM & IEEE, 2013, 2015). Such taxonomies and BOKs represented an important evolution of software engineering as an engineering discipline but in isolation led educators to believe content coverage is more important than pedagogy. A rising trend in engineering programs towards hands-on or experiential learning (Sheppard et al., 2008; NAE, 2005) produces learners more engaged than those in traditional lecture-oriented classes. Our program's industry focus, combined with pedagogical ideas emerging in engineering education, influenced a new innovative design of the curriculum.

The BSSE and MSSE programs moved away from the curricular patterns suggested in the literature (ACM and IEEE, 2004) and focused on a project-centered delivery of core content. Instead of lecture-oriented courses, such as Requirements Engineering, Design, and Verification and Validation followed by a culminating capstone project commonly found in software engineering curricula, the new degree program design offered a series of project experiences incorporating these content areas. The design goal was to avoid disjoint project experiences of the freshman (first-year) project and senior (fourth-year) capstone (Shepard, 2001) and instead use project-based implementation as the main delivery mechanism. The Software Enterprise at ASU is an innovative effort to respond to these challenges using project experiences as the contextual teaching and learning vehicle. At the course level it defines a delivery structure that integrates established learning techniques around a project-based contextualized learning experience. At the degree program level, the Enterprise constitutes a professional spine (Sheppard et al., 2008), weaving project sequences throughout degree program delivery, integrating against program outcomes at each year of the major (Gary et al., 2013).

Figure 2 shows the professional spine curricular design pattern of the BSSE at ASU. Reading from bottom-to-top, 1st- and 2nd-year undergraduate students are exposed to traditional computing content (CS I&II, Languages, Data Structures and Algorithms), a freshman engineering design experience (FSE100), math and lab science including discrete mathematics (not shown), and a liberal arts foundation through university general studies requirements (not shown). In the 2nd semester of the 2nd year, students enter the Software Enterprise project sequence (SER216) and continue this sequence through

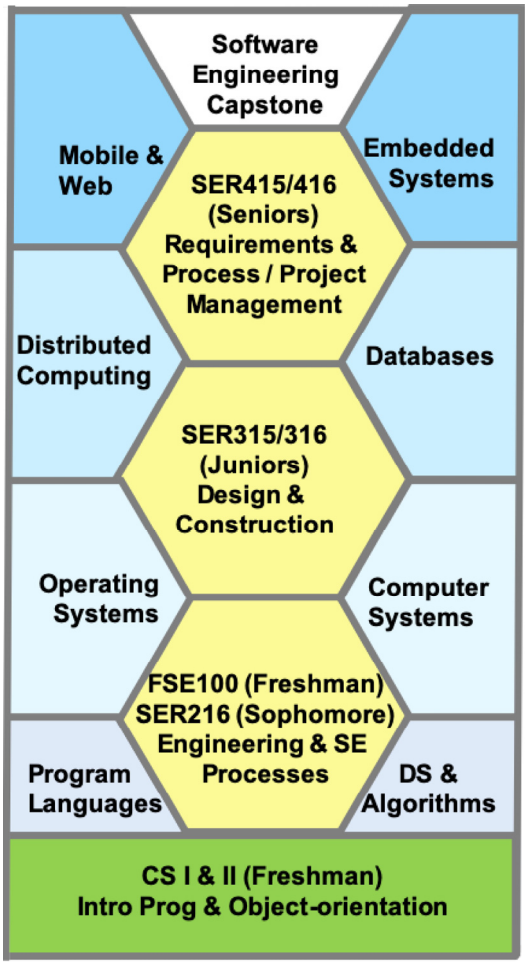

FIG. 2: The *project spine* curricular design of the BS in software engineering at ASU. The Software Enterprise project sequence courses form the spine, supported by traditional computing foundations courses and integrating advanced computing concepts in upper-division years.

degree completion. At each instance, there is a software engineering lifecycle process emphasis but also an integrative feature based on the advanced computing (Operating/Computer Systems) and software engineering (Distributed Computing, Databases) upper-division course requirements. In the 2nd semester of the 3rd and 4th years, students may choose electives such as Web and Mobile Applications or Embedded Systems.

Starting in 2014, ASU began offering the BSSE through ASUOnline. The online offering is exactly the same as the on-campus offering, providing access to the same courses, under the same program outcomes and major map requirements. This includes the Software Enterprise project sequence, which is more challenging to deliver in an online environment (Gary et al., 2017). Since the inception of the online offering, the program has seen rapid growth (Fig. 1). While there has been an increase in faculty hires, enrollment growth has

been so rapid that it outpaces teaching capacity growth. The necessity to operate at scale and deliver a rigorous degree program are the driving forces in the need to innovate with technology and pedagogy.

In our prior work (Gary et al., 2017) we enumerated specific concerns from the faculty at the time of inception of the online program in ramping up a large online engineering degree program. In the ensuing three years the faculty and the administration have taken aggressive steps to address these concerns and are now focusing on continuous improvement and excellence in online education. In this transition, we instituted internal quality process initiatives to promote continuous improvement and emphasized pedagogical innovations with the targeted application of technology to enhance student feedback and foster a community of scholars with online students and faculty.

4. QUALITY IMPROVEMENT PROCESSES

In the first three years of the online program offering, the faculty were primarily responsible for almost all aspects of delivery: recording course content, facilitating interaction and collaboration, and gathering assessment data for an impending ABET evaluation process. During this period the initial strategy was to 1) replicate on-campus course offerings online and 2) rollout course offerings incrementally one year at a time. Both decisions were significant. Faculty debated a strategy of creating distinct online courses and a distinct major map (degree plan) due to concerns about the ability to deliver a project-based spine degree design via the Software Enterprise in an online setting (see previous section). However, the faculty ultimately decided to replicate the course offerings and major map, with the exception of a few upper-division electives, as the project-based approach is considered core to program values. The decision to roll out incrementally (one year at a time) was seen as a way to amortize the workload and gain early feedback over a period of time (4 years); however, the rapid increase in enrollment in the program together with an influx of transfer students forced the faculty to accelerate online course development. The result was a short period of intense course development of project-based courses with little opportunity for feedback in the process, nor for the administration to roll out impactful services institution wide.

As an early adopter of online offerings at ASU, the full range of training, recording, media editing, and other support services were not fully matured and available to faculty. The resulting courses were of “good enough” quality. However, later concerns arose around the scale and adaptability of the online curriculum—How frequently did course content need to be revised? What supplementary guides needed to exist for different faculty and adjunct instructors? How can quality and consistency be maintained across all courses?

At the end of these years the faculty, and the administration, began to understand online delivery at scale better. For the administration's part, multiple service offerings were matured and supported at all campuses; further, the Engineering Dean's office created its own local support structure for online offerings. These changes made instructional designers, media specialists, and recording studios available locally on the Polytechnic campus, eliminating the need to travel to a remote campus for such services. Importantly, a model was created for the periodic refresh of course content—something that is critical in a fast-moving discipline like software engineering where the tools and even the primary concepts change more frequently than more mature engineering disciplines.

For the faculty's part, the rapid acceleration of online enrollment growth (Fig. 1) led to concerns about the ability to provide a consistent quality experience for students. While the first iteration of course development resulted in "good enough" courses, these courses often employed their own approaches to course navigation, student interaction, rollout of course content, and collaboration expectations. While these aspects also exist in on-campus courses, where the "instructor is king," inconsistencies in online delivery leads to gross inefficiencies in the student experience. Online students have more difficulty, without the benefit of a synchronous meeting time and typically with more outside personal and professional obligations, dealing with the dissonance that comes from having to locate information and conduct communication with different online tools and navigation structures.

The program chair for SE formed a special Faculty Working Group (FWG) which evaluated the online course offerings using a newly created a set of rubrics. The initial set of rubrics produced by the FWG is given in Table 1, and the detailed findings of the FWG are presented in Gary et al. (2017). As a growth of the original working group course review process, the faculty have created a more comprehensive and general course review process. The aim is to support continuous improvement of the courses within the program, the interactions between those courses, and how they support program outcomes. This supported a better understanding of the original implementation of the online program and resulted in a set of recommendations to the school. However, the fundamental limitation of the work by the FWG was that it was performed as a single pass to understand the state of the program, and the specific results collected do not reflect the current program. Individual courses, as well as the degree pathway, have evolved over time due to program needs and faculty input, as well as advising input. Another limitation in the FWG process was treating course development and teaching (an "execution" of a course) as being intertwined. This is reasonable for an on-campus course. However, due to the growth of our online program and the desire to use prerecorded materials to scale course offerings, these aspects of course evaluation may be decoupled. The faculty member responsible for

overseeing standards in a course (the coordinator), the one responsible for recording lectures or creating assignments (the developer), and the one responsible for teaching during a semester (the instructor) may all be different faculty. Furthermore, the program is moving towards a model of shared course stewardship where, for instance, a course may contain content developed by several different instructors according to expertise.

TABLE 1: Rubrics for online course shell evaluation. Additional rubrics are specified for Communication, Instructional Resources / Content Delivery, Academic Integrity and Quality, Student Support, and Course Administration.

	Criteria	Missing	Developing	Accomplished	Innovative
LOs (LOs) & Alignment to POs (POs)	Measu-rable course LOs	> 75% of course LOs not measurable	> 50% of course LOs not measurable	> 25% of course LOs not measurable	< 25% of course LOs not measurable
	Course LO construc-tion	Course LOs not provided.	> 50% of the course LOs do not properly use suitable LO construction verbs and levels, such as Bloom's	> 25% of the course LOs do not properly use suitable LO construction verbs and levels, such as Bloom's	< 25% of the course LOs do not properly use suitable LO construction verbs and levels, such as Bloom's
	Measu-rable module LOs	> 75% of module LOs not measurable	> 50% of module LOs not measurable	> 25% of module LOs not measurable	< 25% of module LOs not measurable
	Module LO construc-tion	Module LOs not provided for 75% or more of the modules.	> 50% of the module LOs do not properly use suitable LO construction verbs and levels, such as Bloom's	> 25% of the module LOs do not properly use suitable LO construction verbs and levels, such as Bloom's	< 25% of the module LOs do not properly use suitable LO construction verbs and levels, such as Bloom's
	Course alignment with POs	Alignment of course LOs to POs not provided	> 50% of course LOs do not align to POs; either missing or incorrectly specified	> 1, but < 50% of the course LOs do not align to POs; either missing or incorrectly specified	All course LOs align to POs (present and correct)

	Criteria	Missing	Developing	Accomplished	Innovative
	Module alignment with course LOs	Alignment of module LOs to course outcomes not provided	> 50% of modules have LOs that do not align to course outcomes; either missing or incorrectly specified	> 1, but < 50% of the modules have LOs that do not align to course outcomes; either missing or incorrectly specified	All modules have LOs aligned to course outcomes (present and correct)
Navigation & Presentation	Adhe-rence to course-level standard template	Missing any of Welcome & Start Here, Staff, Schedule, & Discussion in upper-left nav	Links to course modules available but not navigable in a week-by-week fashion	Required links present but supporting links to Technology/ Resources, Announce-ments, and Assignments not present	All links specified in previous categories present in the left nav
	Course Schedule	Course Schedule is present in left nav	Schedule shows a progression of content	Week-by-week schedule shown in left nav	Schedule of content coverage follows a topic map showing how content is related to each other
	Adhe-rence to module-level template	Modules follow different organiza-tional schemes, making it difficult to understand content flow	Modules follow a consistent organizational scheme but LOs and an activity summary for the module are not at the top of the page	Modules follow a consistent organizational scheme with LOs at the top of the page	Modules include a summary of the activities (what a student needs to do) near the top and may make use of adaptive release features

	Criteria	Missing	Developing	Accomplished	Innovative
	Functio-nality of tools and links	Course contains broken links or tools that render it unusable	Course links and tools functional but complicated to use effectively and detract from learning	All tools and links operational and do not detract from instruction	All tools and links operational and enhance instruction. Help or tech support resources given for all tools used within the course

LO – learning outcome, PO – program outcome

The software engineering program's standing undergraduate program committee (UPC, which makes recommendations to administration regarding the program and courses) proposed a revised review process to evaluate courses which has three aspects: 1) alignment, 2) quality, and 3) consistency. The ***alignment*** aspect aims to capture how well program outcomes decompose to course description and course outcomes, and then down to course assessments. This serves two important purposes: ensuring the course's specification is well defined at each level of instructional outcome and ensuring traceability from program outcomes to assessments. The ***quality*** aspect focuses on aspects of the course related to the experience of both taking and teaching a course. Quality is used to refer broadly to success in a course: students meet course outcomes after its completion. From a student perspective, a quality course is one that provides features like course policies that are clear, instruction that is complete and understandable, assessments that are aligned to instruction, reasonable workload, reasonable deadlines, useful feedback from assessments, provides appropriate communication channels, supports accessibility, and so on. From an instructor perspective, a quality course is that one that provides features like instructions on how to run a course, a complete set of instruction and assessment resources, reasonable instructor workload, a grading/TA workload that aligns with department resourcing, assessments that support measurement of outcomes, infrastructure to address academic integrity, and so on. The ***consistency*** aspect views a course in terms of reproducibility across semesters, where instructor activities and specific course content may vary, and with respect to a baseline experience. Courses should demonstrate reproducible results (e.g., student meeting outcomes) across similar cohorts. To gain traceability on consistency and provide a pathway for correction, this aspect also tracks provenance. Although a course may be taught from a standard template, changes still occur during execution as an instructor responds to the unique cohort. Changes also

occur to the standard template. To draw conclusions about the state of courses running in a semester, the relationship between those courses and the template courses reviewed must be known.

When reviewing alignment and quality, faculty evaluate the master course shell (an LMS representation of a course). A master shell represents a standard template for a course offering that is maintained by a course coordinator and developer. This methodology is enabled by the online program, which provides a reliable platform to perform course evaluations, as time for course development is resourced independently of running them, and each semester generates a set of traceable artifacts (both instructional and assessment results) which are archived implicitly at low cost-to-faculty time by an LMS. Reviews on consistency use the course shell associated with each semester, which contains the additional assessment results. A review is conducted by two faculty members (typically one with domain experience and one without) and the input of the appropriate faculty (coordinator for alignment, developer for quality, coordinator and instructor for consistency) to facilitate accuracy and objectivity. Typically, one aspect of a course is reviewed at a time, according to a simple process. As an example, we use the following process for alignment reviews:

1. The course coordinator provides documents (e.g., syllabus, course outcomes to assignment mappings) to the reviewers (or as links to the master course shell).
2. Two faculty members review the provided document and course shell to fill out a rubric.
3. The reviewers meet with the course coordinator to discuss questions that arose during the review. Reviewers and coordinator briefly discuss how well the current rubric captures the course being reviewed and make suggestions for rubric improvement.
4. Based on the findings of the reviewers, a set of improvements may be communicated to the course coordinator.
5. If significant issues are found, change recommendations are made to the course coordinator and a follow-up meeting is scheduled to review changes.
6. After the follow-up, if there are still severe issues, the program chair is notified.

The different parts of a course are evaluated using a binary rubric. The rubrics are lightweight to enable accuracy and ease of evaluation while exposing concerns to external stakeholders. The result of a review is a completed rubric which is archived with any additional course documentation collected from the developer. Items within a rubric are ranked by severity; this enables a summarization and exposure of important information

collected during a review. The rubric results (consisting of rubric values and rubric versioning information) are recorded in a course status tracking spreadsheet.

Over time the ability of the recorded reviews decays as courses develop past the state in which they were reviewed. This may impact both evaluation aspects—alignment and quality. A course that is being tracked may have its evaluations "expired" by several mechanisms. For instance, department policy may dictate syllabus content needs to change (*alignment;* resulting in the need to update one rubric item) or a course may need to be refreshed (*quality;* an entirely new review needs to be performed). In contrast, consistency reviews do not expire but rather are performed every semester on a sampling of courses. Deviations between semesters, or from the baseline, are taken as potential issues with the course.

Another aspect of course reviews are instructor-led reflections based on course data. The existing process in the SE program is to use faculty course assessment reports (FCARS) (Estell, 2007). FCARS ask instructors to classify student cohort performance as Excellent, Adequate, Minimal, and Unsatisfactory, and provide traceability of course modifications and assessment of course outcomes to program outcomes. The FCAR reporting instrument was updated to specifically document issues and improvements to online courses, with an emphasis on student interaction with the course. Additionally, the standard course evaluation form given to students at the end of each session was augmented for online students with a section specific to online delivery.

Of course, peer review of course shells and instructor-led self-assessment does not help if a vehicle is not in place to enact change. In some cases, minor improvements to course content are handled as part of a course coordinator's typical teaching load. However, more significant refreshes to content are now part of a regular RFP-style process where course shells undergo redevelopment at least once per accreditation cycle, and in cases of fast-moving topics or new content, perhaps more frequently. Further, these evaluations are used as the basis for requesting special resourcing for certain classes, such as project-based classes or classes that have demonstrated a significant deficiency or bottleneck in student retention and success.

This continuous improvement quality evaluation process ensures that online courses are delivered in a consistent manner no matter who is adding or modifying content and who is delivering the course. This ensures both quality and flexibility to add instructional resources at scale. Further, as an ABET-accredited program, it is critically important that all course offerings remained aligned with program outcomes. With the recent changes to ABET engineering criteria (ABET, 2018) (Criterion 3 Student Outcomes in particular), the BSSE program updated program outcomes and needed to align existing and new course offerings to support these new program outcomes. The quality improvement processes

described in this section ensure course offerings remain aligned with program outcomes even in the face of significant growth.

5. INNOVATIONS TO IMPROVE ONLINE ENGINEERING EDUCATION

The challenges of transitioning an existing engineering program to an online modality we view in three dimensions. The most obvious dimension that comes to mind as an educator when moving a degree program online is the effort involved in the transition. In the BSSE at ASU, this concern is compounded by the project-centric nature of the Software Enterprise spine and an accelerated delivery timeline (ASU online courses are typically offered in 7.5-week sessions compared to traditional 15-week semesters for on-campus delivery). A not as obvious dimension derives from the recognition that online students need a community just as much, if not more, than on-campus students. The typical online student, working alone at home, needs to feel connected to the university environment. Certainly, some of this derives from basic classroom support needs, but we find a deeper interaction is needed for students to feel connected to peers and the university as a whole, to allow support for common experiences (and frustrations). Add to this the need to scale this interaction to over a thousand students, who in the absence of community resort to a 1:many communication model proportional to the student-to-faculty ratio of the program. The final dimension is a lack of recognition of online education as a core aspect of the university. Initially, at Arizona State University online delivery was viewed as an “add-on” by faculty and administration with a simple “record and replay” model assumed by default. After our first couple of years of online delivery, we quickly found that to be effective, we had to reexamine our approach to curriculum delivery and student support throughout all of our processes. This section will present strategic innovations in pedagogy, the targeted application of technology, and faculty internal quality improvement that we have implemented to address the essential complexities† of scalability and pedagogical challenges inherent in online engineering education.

5.1 Pedagogical Innovations

The Software Enterprise (Gary, 2008; Gary, 2009) is an innovative pedagogical model for accelerating a student’s competencies from understanding to comprehension to applied knowledge by colocating preparation, discussion, practice, reflection, and contextualized learning activities in-time. In this model, learners prepare for a module by doing readings, tutorials, or research before a class meeting time. The class discusses the module’s concepts in a lecture or seminar-style setting. The students then practice with a tool or technique that reinforces the concepts in the next class meeting. Reflection completes the cycle, internalizing concepts and validating student expectations, or hypotheses, for the utility of the concept. Then, students apply the concept in an ongoing team-oriented,

scalable project and reflect again to (in)validate their earlier hypotheses. The Software Enterprise is a specific pedagogical instance of Kolb's experiential learning cycle (Kolb, 1984).

At the onset of the online program in 2014, the on-campus program was just graduating its first cohort of students and planning for an anticipated initial ABET accreditation visit in 2015. An immediate concern was whether we could continue to deliver a project-centric approach to software engineering education, or whether we had to fall back to a more traditional delivery for online and teach to the same program outcomes or modify the on-campus delivery to also use a more traditional delivery model. The faculty decided to move forward online with the same project-centric offering as the on-campus program, as it is core to our philosophy regarding engineering education as a collaborative endeavor, and such communication and organization skills are tantamount to the profession. However, team-based project-centric learning in an online setting presents significant challenges. The experiential learning model means that student teams face a steady stream of new content, new practice on the skills presented in that content, project-based development activities, and reflection. The compressed time period of course delivery (7.5 weeks instead of 15 weeks) and the asynchronous nature of ASU's online model put additional pressures on this model. Specifically, the model requires significant interaction between students and between students and instructional staff, and necessitates scalable, rapid feedback (formative and summative). We have utilized technology to address these requirements. To elaborate on our innovative applications of technology, the next section describes the transformation of the most intensive project course in the Enterprise sequence.

5.2 SER316: Intensive Project-Centric Learning at Scale

SER316, Software Enterprise: Construction and Transition, is the fourth course in the Enterprise sequence typically taken in the 2nd semester of the 3rd year of the undergraduate program. Students in this course learn best practices in quality software construction (unit testing, static analysis, metrics, code reviews, refactoring, etc.) and transitioning to operation (source code control, change management, build and deployment processes, release management, etc.). Teams employ the Scrum (Schwaber and Beedle, 2002) agile methodology as their software development lifecycle process (SDLP). They are given a relatively large (~23k lines of code, which is large to students at this level) preexisting codebase for an application and asked to implement new features and improve internal quality. The past three primary (Spring semester) offerings have had between 82 and 113 students, typically organized in teams of 4–5 students. Marshaling this many student teams through an agile process in a 7.5-week period is a significant

challenge, and we use it to represent the application of technological innovations in the software engineering program.

In software engineering, tooling is heavily relied upon for managing software processes and software artifacts. Individual integrated development environments (IDEs) are typically a choice preference of individual developers in industry. We require students to use Eclipse (an IDE) to support integrations of common low-level development practices through plugins, though this choice has no pedagogical or professional development import other than efficiency in supporting a common platform at scale. Tools that support team coordination and communication, source code change management, and software quality are critical in collaborative professional practice for engineering organizations to deliver software products of acceptable quality at breakneck speed. Such shared tools are used by the software development organization as a whole to utilize on all projects. The primary decision point for choosing these tools is the SDLP employed by the organization, and the most prevalent SLDP methodology is Scrum, one of the agile family of software methods that emphasizes speed, collaboration, and reacting to change. The BSSE program was an early adopter in academia of agile methods and has evolved the collaborative project support tools it requires students to use based on Scrum.

To support project teams in the Enterprise pedagogical model in SER316, we have adopted a number of off-the-shelf software engineering (OTS-SE) tools and developed several custom solutions. The OTS-SE tools should represent modern industry practice but also support open integration and data reporting through programmatic interfaces (APIs). The OTS-SE tools we currently include are Eclipse and associated plugins for skill practice, Git/GitHub for source code control, Taiga for Scrumboard‡ support, and Travis-CI for continuous integration and testing. We created custom tools that push/pull from the various OTS-SE APIs for features such as team project formation, continuous formative feedback, and autograding of certain project rubrics.

For team project formation, we initially employed the CATME platform from Purdue University (Layton et al., 2010). However, CATME is a closed platform with no open API access that switched to a license subscription model, so we created a custom tool called Nicest (Gary et al., 2018) and recently scripted a new version that incorporates more information regarding online students, as scheduling based on work commitments and time zones has become the predominant complexity factors. We ask students to submit a survey with information about their knowledge, time zone, available times, and for preferences who they would like to work with and who they would prefer not to work with. Our tool assembles this data and sorts students by time zone and preference so the forming of groups can be partially automated. Due to sensitivity to particular student situations (students in the military, students on individualized learning plans, etc.), manual

student assignment to teams is still performed at the end. After teams are formed, private GitHub repositories and an initial Taiga Scrumboard is created for each project instance by a push to the respective APIs through our tool. It sets the master branch on GitHub as protected, so code reviews are needed before a merge is allowed and creates an initial sprint on the Scrumboard with some *user stories* (agile software requirements) for the students to get started. This gives the students the correct tools to get going in the project and takes away the burden of the instructor to set things up manually. Through this team project formation tool this one-time provisioning process happens quickly and at scale for a large number of teams.

The time pressures and scale of the online program make providing frequent formative and graded feedback challenging. We have adopted an approach we refer to as *continuous assessment* (Gary and Xavier, 2015; Ghiatău et al., 2011), emphasizing the need for both formative and summative feedback “in-time” so students may incorporate this guidance into present iterations of the Enterprise experiential cycle (Crisp, 2007). For on-campus instances of SER316 and other project courses, class time is reserved for teams to meet and for the instructional staff to informally meet with teams to offer guidance (formative) and to provide more formal review milestones which are scored (summative). Only minor parts of this process may be replicated online, for example, short YouTube video presentations for formal review milestones. More critically, it is very difficult to provide timely formative feedback to guide students and teams through the context-oriented application of new software construction skills. We view this formative feedback as essential to deeper understanding of the engineering process, where sound engineering judgment is often applied to determine the best among a number of possible actions. Software engineering teams are continuously faced with the question of “How much of *X* do I perform to ensure sufficient quality in the face of time pressures?” where *X* is a quality injection practice, such as code reviews, unit testing, or refactoring. We view this as somewhat unique to software engineering practice compared to other engineering disciplines, where quality specifications on “tangible” deliverables may be expressed more exactly as quantifiable constraints. Teaching students how to make professional software engineering judgments in such a context is a critical learning outcome of our degree program.

We have iterated over the design and implementation of custom tools to assist with (primarily) formative and summative feedback in project-centric courses. Descriptions of previous iterations of some of these tools have been published under the Continuous Assessment Platform (Gary and Xavier, 2015; Xavier et al., 2016; Gary et al., 2018); here we describe the latest generation of these tools. A GitHub scraper tool shows when someone in the team committed new or updated code to their Git repository, how many

changes s/he made, and the corresponding comments. This gives as a fast and easy overview if students committed frequently and consistently (as we want for Scrum), and we see if they are able to write good commit messages and what they have been working on. Through a link we can go to the commit directly on GitHub and check these artifacts and enter our own comments. For Taiga, we created a tool in Python that analyzes a team's Scrumboard and gives us information about the current sprint (agile term for short iteration). With this tool we get a report on requirements completed ("story points" in agile terminology), worked on, and not completed, work items created and completed, and patterns in how these work items change state (Fig. 3). This provides a good overview about the team dynamic and how well they are applying the Scrum process.

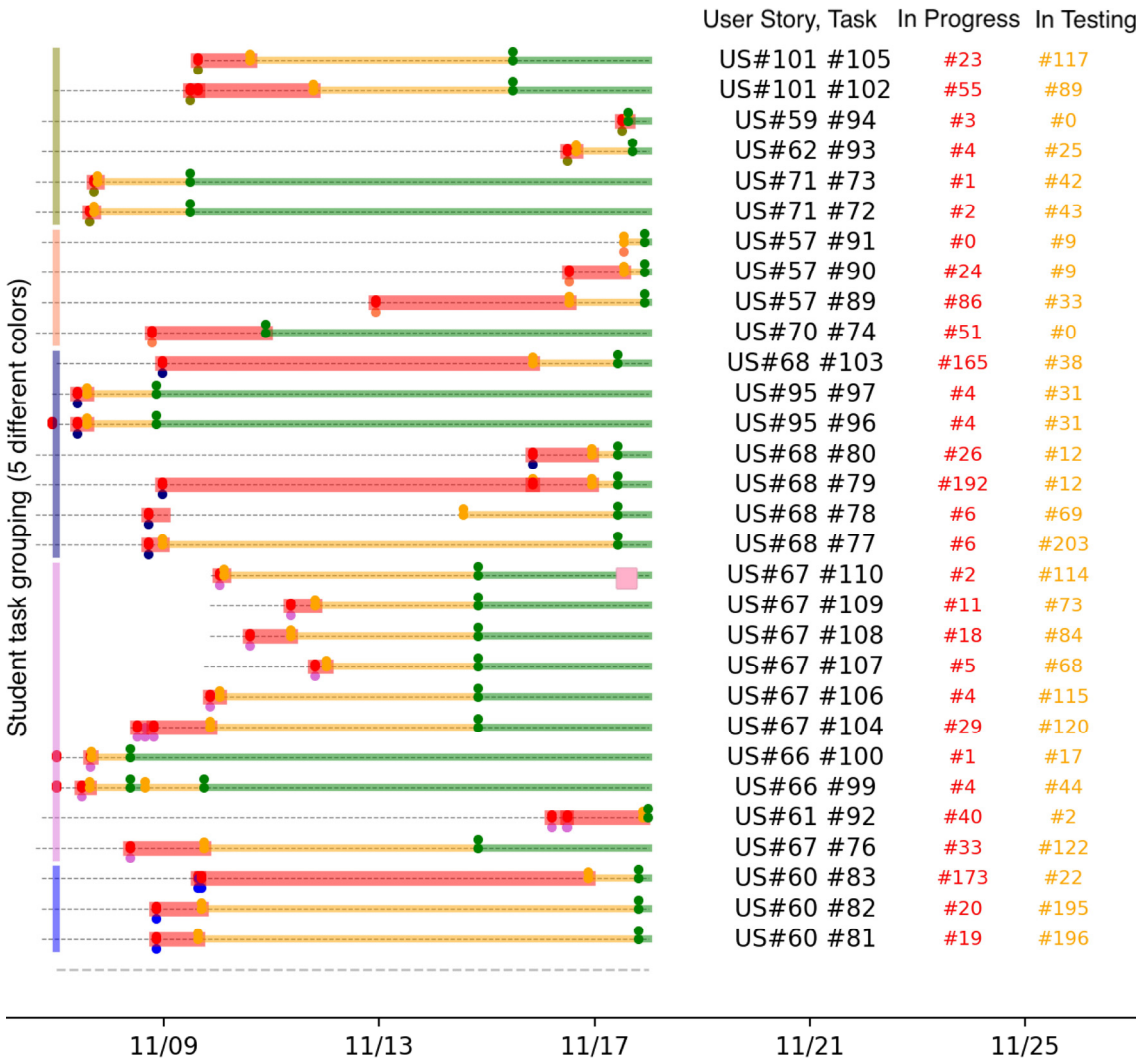

FIG. 3: Example report from the feedback tool. Vertical lines represent different students. Each horizontal line represents one Task (– Lifetime of a task, red – In Progress, yellow – In Testing, green – In Done), after each line the US number and task number is presented, the red number represents the hours the task was In Progress, and the yellow how long In Testing.

5.3 Peer and Instructor Communication

A somewhat unexpected consequence of porting a program to an online modality is the need to foster an extensive online community. Initially our concern was focused more on producing quality course content than student-student and student-faculty communication (Alqurashi, 2019). We hypothesize that this is partly due to the prevalence of the graduate-level, specifically master's programs (including some at ASU), we reviewed. Undergraduate students require more interaction, and more importantly, a sense of community (Arasaratnam-Smith and Northcote, 2017; Muljana and Luo, 2019; Hodgson and McConnell, 2019). The on-campus undergraduate student certainly acquires a significant amount of process knowledge from peers, and in retrospect we should have identified this as a concern much earlier.

Initially, faculty used standard communication tools provided by the Learning Management System (LMS) for the course, namely, email and threaded discussion forums. Email is problematic in that it fosters a one-to-one communication model between student and instructor. Threaded discussion forums are better, though they lack a real-time conversational quality, and many students are reticent to post questions in open forums. In search of better solutions, the faculty piloted the use of Zoom videoconferencing and Slack workspaces. Videoconferencing solutions are of limited scale and utility in an online program that advertises asynchronous delivery, but we found there is a percentage of students who do like to interact directly with instructional staff for office hours and for professional mentorship. While there are many videoconferencing solutions on the market, Zoom was chosen as it scales well with many students in a virtual room at once, enables screensharing, cloud-based recording, dashboard analytics reporting tools, and an API that allows for custom data gathering and reporting. Shortly after the SE program piloted Zoom, ASU decided on Zoom as the videoconferencing platform of choice, acquiring an institutional license.

Slack has had the greatest impact on student-faculty and student-student communication. In a previous paper (Mehlhase et al., 2019) we described experiences piloting Slack in the classroom, as do others in the literature (Cyders and Hilterbrane, 2016; Fulton, 2017; White et al., 2017). Slack provides online workspaces that can be organized in channels for different types of conversations. Just as the faculty arrived at a common navigational pattern for course shells in the LMS, we also arrived at a common set of channels in a course instance Slack workspace. In this way students can navigate from workspace to workspace and know which channel to ask what types of questions, and additional channels may be used to encourage more informal conversations or discuss issues broader than the course topics, such as professional development questions. We credit

the online SE students for introducing Slack, as they created the first online community as a sanctioned student club and began to invite faculty *ad hoc* to participate. After a year of informal utilization of Slack via personal accounts, the SE program piloted the Slack Enterprise Grid for ASU, which has since adopted the platform for both online and on-campus communication.

The Slack Enterprise Grid provides some outstanding features, such as channels, threaded discussions inline, “pinning” responses for later referral, an analytics dashboard, and an open API for custom data reporting. There are drawbacks, however, such as the expectation that instructional staff is available 24/7. Another issue with an active Slack workspace is that it gets really crowded in the channels very fast, which increases the amount of communication but makes it hard for students to find answers fast. To work against this drawback, we created a simple app that helps us mark important questions and answer pairs and move them into a Frequently Asked Questions (FAQ) channel (Fig. 4).

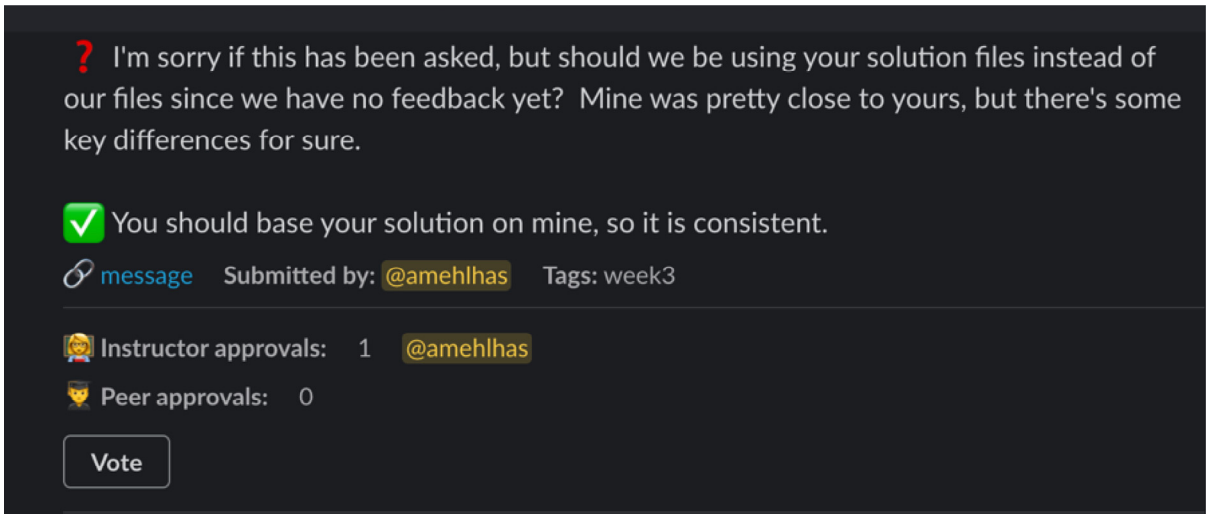

FIG. 4: A Q&A entry in the Slack FAQ channel showing a question and answer, a link to the original thread in Slack, and who submitted it and related tags. We also integrated a voting system so the instructional team can moderate Q&A and students can vote for items.

Communication is at the heart of team-based project-centered learning and agile software engineering methods. While most aspects of the Enterprise pedagogical model are targeted for individual students, the project-based contextual activities are team-based. Also, our experience with Enterprise courses over more than a decade of practice is that individual student expectations and reflections are influenced by their teams. Therefore, it was very important to incorporate Slack as a medium for communication for project-based teams in the Software Enterprise. For communication in teams we set up Slack private channels for each team, and one “group general” channel for their project communication.

The Slack Enterprise Grid has many apps ("bots") and integrations available, such as Polls, Daily Standup (a Scrum method daily meeting), and GitHub/Travis-CI/Taiga integrations to help them stay organized. Another channel "group instructor" is created for each group in which the team can directly communicate with the instructor, so the whole team can ask relevant questions and see the answer. This helps the instructional staff to make sure that the whole team is informed and not just one student out of the team. This communication has worked really well for our project groups.

Finally, we note that asynchronous communication in the form of short videos on a weekly basis summarizing common class questions and providing class-wide feedback and encouragement has also had a positive effect, as reported in Mehlhase et al. (2019).

6. INNOVATION IMPACT

The innovations the software engineering faculty have introduced have shown a demonstrable difference in the quality of the program online offering. Students have taken notice. Table 2 shows the year-over-year course evaluations for the degree program, both overall and just online course offerings. After showing a decline in course satisfaction during academic years 2014–15 through 2016–17, the evaluation scores since implementing the quality improvements described in this paper consistently trend up (academic years 2016–17 through 2018–19). Further, outside of a single semester (Spring 2018), the delta between on-campus and online course evaluation overall scores is narrowing, suggesting the improvements are addressing systemic challenges in replicating the traditional college experience. The online components of course evaluations also show improvement. For online courses, students are asked questions specific to the online environment and asked to rate the overall online experience. Table 3 shows the comparisons between the first instances of an online course and the most recent instance. Again, a sizable increase is shown in each respective semester.

TABLE 2: Summary of course ratings from student course evaluation surveys. Summaries are presented by Fall and Spring semester for an equitable comparison between courses and student position in the major map.

Semester	Online	Delta to on-campus	Semester	Online	Delta to on-campus
Fall 14	3.63	(0.49)	Spring 15	3.66	(0.34)
Fall 15	3.26	(0.53)	Spring 16	3.6	(0.18)
Fall 16	3.22	(0.6)	Spring 17	3.57	(0.19)
Fall 17	3.46	(0.36)	Spring 18	3.52	(0.32)
Fall 18	3.58	(0.19)	Spring 19	3.75	(0.12)

Ratings decline from the 2014–15 academic year, then mostly increase thereafter

TABLE 3: Average online component overall rating score comparison from the first offering of a course to the most recent

Semester	Offerings	Rating	Semester	Offerings	Rating	Delta
Fall 15 or 16	7	4.07	Fall 18	9	4.21	0.14
Spring 15 or 16	9	3.73	Spring 19	9	3.99	0.26

First offerings included are in the Fall 2015 to Spring 2016 time period as courses were rolled out incrementally. Most recent offerings were all in the 2018–19 academic year. Trending data is not given as course refreshes occur at different times, though each course included has undergone at least one refresh between the first and most recent offerings.

Of course, online course evaluations are not scientifically rigorous, and these comparisons have not been controlled for limitations such as the type and experience of instructor, class size, and other factors. However, we argue it demonstrates consistent improvement despite variability outside the unit's control—enrollment, new technology platforms, use of different instructors in different semesters. Given the large number of students and the significant challenges teaching online at scale, we consider these results a success, while acknowledging there remains room to improve.

We note anecdotally that quality improvements in online delivery have also led to improvements in on-campus delivery. The continuous process improvement efforts described in this paper have had a positive effect on all course instances. The rigor of the evaluation process has improved all three evaluative aspects (alignment, quality, and consistency). For example, *consistency* has been improved by ensuring rotating instructors (we sometimes employ faculty adjuncts to come in and lead a course) cover the same content and address the same course learning outcomes. Further, the availability of media assets has enabled several on-campus courses to be "flipped," making more efficient use of classroom contact time. Finally, the application of technology to support formative feedback and better communication benefits all students, not just those online.

We would be remiss for not acknowledging the role of the students themselves in the quality improvement process. Above we described how the students started using Slack and introduced it to the faculty; the students have also initiated their own surveys, recommending, for example, shorter and custom short videos, faster feedback, use of Slack over discussion forums—all themes discussed in this paper.

REFERENCES

Accreditation Board for Engineering and Technology (ABET), *Criteria for Accrediting Engineering Programs*, accessed September 30, 2019, from https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2018-2019/, 2018.

Alqurashi, E., Predicting Student Satisfaction and Perceived Learning within Online Learning Environments, *Distance Ed.*, vol. **40**, no. 1, pp. 133–148, 2019. DOI: 10.1080/01587919.2018.1553562

Arasaratnam-Smith, L. and Northcote, M., Community in Online Higher Education: Challenges and Opportunities, *Elect. J. e-Learning*, vol. **15**, no. 2, pp. 188–198, 2017.

Association for Computing Machinery & Institute for Electrical and Electronic Engineers Computer Society, Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, Joint Task Force on Computing Curricula, 2004.

Association for Computing Machinery & Institute for Electrical and Electronic Engineers Computer Society, Software Engineering 2014 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, Joint Task Force on Computing Curricula, 2015.

Association for Computing Machinery & Institute for Electrical and Electronic Engineers Computer Society Joint Task Force, Computer Science Curricula 2013, 2013.

Beetham, H. and Sharpe, R., Eds., An Introduction to Rethinking Pedagogy, in *Rethinking Pedagogy for a Digital Age*, New York: Routledge, 2020.

Bourque, P. and Fairley, R.E., Eds., *Guide to the Software Engineering Body of Knowledge, Version 3.0*, IEEE Computer Society, 2014.

Brooks, F.P. and Bullet, N.S., Essence and Accidents of Software Engineering, *IEEE Comput.*, vol. **20**, no. 4, pp. 10–19, 1987.

Bureau of Labor and Statistics, *Employment Outlook 2018–28*, accessed September 30, 2019, from http://www.bls.gov, 2019.

Cockburn, A., *Agile Software Development: The Cooperative Game*, London: Pearson Education, 2006.

Crisp, B.R., Is It Worth the Effort? How Feedback Influences Students' Subsequent Submission of Assessable Work, *Assess. Eval. Higher Ed.*, vol. **32**, no. 5, pp. 571–581, 2007.

Cyders, T. and Hilterbrane, A., Classroom Integration of the Slack Team Collaboration Tool, in *Proc. of the 2016 National Capstone Conf.*, Columbus, OH, June 2016.

Estell, J.K., Streamlining the Assessment Process with the Faculty Course Assessment Report, Workshop, Frontiers in Education, 2007.

Fulton, L., Slack in Education: A Case Study of Alternative Communication for Groupwork in Graduate Level Online Education, in *Proc. of Society for Information Technology and Teacher Education Int. Conf.*, Savannah, GA, pp. 1458–1463, April 2017.

Gary, K., The Software Enterprise: Practicing Best Practices in Software Engineering Education, *Int. J. Eng. Ed.*, vol. **24**, no. 4, pp. 705–716, 2008.

Gary, K., The Software Enterprise: Preparing Industry-Ready Software Engineers, in *Software Engineering: Effective Teaching and Learning Approaches*, H. Ellis, S. Demurjian, and J.F., Naveda, Eds., Los Angeles: Idea Group Publishing, 2009.

Gary, K. and Xavier, S., Agile Learning through Continuous Assessment, *Proc. of the ACM/ASEE/IEEE Frontiers in Education Conf. (FIE'15)*, El Paso, TX, October 2015.

Gary, K., Lindquist, T., Bansal, S., and Ghazarian, A., A Project Spine for Software Engineering Curricular Design, *Proc. of the 26th Conf. on Software Engineering Education and Training (CSEET)*, 2013.

Gary, K., Sohoni, S., and Lindquist, T., It's Not What You Think: Lessons Learned Developing an Online Software Engineering Program, *Proc. of the 27th Conf. on Computer and Software Engineering Education and Training (CSEE&T)*, 2017.

Gary, K., Johnson, T., Murphy, C., and Athreya, R., Agile Teaching and Learning through Continuous Assessment, *Proc. of the Frontiers in Education of Computer Science Conf. (FECS)*, 2018.

Ghiatău, R., Diac, G., and Curelaru, V., Interaction between Summative and Formative in Higher Education Assessment: Students' Perception, *Soc. Behav. Sci.*, vol. **11**, pp. 220 –224, 2011.

Hodgson, V. and McConnell, D., Networked Learning and Postdigital Education, *Postdigital Sci. Ed.*, vol. **1**, p. 43, 2019.

Kolb, D.A., *Experiential Learning: Experience as the Source of Learning and Development*, Upper Saddle River, NJ: Prentice Hall, 1984.

Lack, K.A., *Current Status of Research on Online Learning in Postsecondary Education*, accessed September 30, 2019, from https://doi.org/10.18665/sr.22463, 2013.

Layton, R.A., Loughry, M.L., Ohland, M.W., and Ricco, G.D., Design and Validation of a Web-Based System for Assigning Members to Teams Using Instructor-Specified Criteria, *Adv. Eng. Ed.*, vol. **2**, no. 1, pp. 1–28, 2010.

Lederman, D., Who is Studying Online (and Where), *Inside Higher*, accessed from https://www.insidehighered.com/digital-learning/article/2018/01/05/new-us-data-show-continued-growth-college-students-studying, 2018.

Means, B., Toyama, Y., Murphy, R., Bakia, M., and Jones, K., Evaluation of Evidence-Based Practices in Online Learning: A Meta-Analysis and Review of Online Learning Studies, U.S. Department of Education, 2009.

Means, B., Bakia, M., and Murphy, R., *Learning Online: What Research Tells Us About Whether, When and How*, Abingdon, UK: Routledge, 2014.

Mehlhase, A., Heinrichs, R., and Gary, K., Effective Use of Slack and Short Video to Scale Online Learning Communities, *Proc. of the Frontiers in Education of Computer Science Conf. (FECS)*, 2019.

Muljana, P.S. and Luo, T., Factors Contributing to Student Retention in Online Learning and Recommended Strategies for Improvement: A Systematic Literature Review, *J. Inf. Technol. Ed.: Res.*, vol. **18**, pp. 19–57, 2019. DOI: https://doi.org/10.28945/4182

National Academy of Engineering, *Educating the Engineer of 2020: Adapting Engineering Education to the New Century*, Washington, DC: The National Academies Press, 2005.

Schwaber, K. and Beedle, M., *Agile Software Development with Scrum (Vol. 1)*, Upper Saddle River, NJ: Prentice Hall, 2002.

Shepard, T., An Efficient Set of Software Degree Programs for One Domain, *Proc. of the 23rd Int. Conf. on Software Engineering*, pp. 623–632, 2001.

Sheppard, S.D., Macatangay, K., Colby, A., and Sullivan, W.M., *Educating Engineers: Designing for the Future of the Field*, San Francisco: Jossey-Bass, 2008.

US News and World Report, *The 25 Best Jobs of 2020*, accessed January 7, 2020, from https://money.usnews.com/money/careers/slideshows/the-25-best-jobs?slide=26, 2020.

US News and World Report, *Engineering Program Rankings*, accessed September 30, 2019, from https://www.usnews.com/education/online-education/bachelors/rankings, 2019.

White, K., Grierson, H., and Wodehouse, A., Using Slack for Synchronous and Asynchronous Communication in a Global Design Project, in *Proc. of the 19th Int. Conf. on Engineering and Product Design Education*, Bristol, UK, pp. 346–351, September 2017.

Wu, D.D., Online Learning in Postsecondary Education: A Review of the Empirical Literature (2013–2014), 2015. DOI: doi.org/10.18665/sr.221027

Xavier, S., Murphy, C., and Gary, K., A Student Activity Dashboard for Ensuring Project-Based Learning Compliance, *Proc. of the Annual Conf. of the American Society for Engineering Education (ASEE)*, 2016.

NOTES:

† Yes, a reference to Brooks's classic delineation of accidental versus essential complexity (Brooks, 1987). ↩

‡ A *Scrumboard* is a process management tool used as an "information radiator" (Cockburn, 2006) for an agile team. ↩