Orchestrating Agile Sprint Reviews in Undergraduate Capstone Projects

Xiaocong Fan
School of Engineering, Behrend College
Pennsylvania State University
Erie, PA 16563
Email: xfan@psu.edu

***Abstract*—This Innovate Practice Full Paper presents an orchestrated way to conducting agile sprint reviews in undergraduate capstone projects. An undergraduate capstone project typically involves a student development team as well as other stakeholders such as faculty advisors and industry mentors. Given that the agile development approach is adopted, a sprint review is conducted at the end of each sprint to demonstrate and critique the features that the development team have completed. Unlike the industrial settings, capstone design students can be challenged when conducting agile sprint reviews. On the one hand, the availability of industry mentors is unpredictable due to their own business commitments and schedules. On the other hand, it is extremely hard to bring all stakeholders together in one meeting. To overcome these difficulties, in our practice we have employed two strategies. First, we used a so-called agile buddy approach which allows each team to acquire an outsider student's opinion on their system. Second, we adopted a web-based collaboration platform such that students can collect critiques asynchronously from different stakeholders.**

***Index Terms*—Capstone projects, teamwork, Agile development, Agile buddy.**

I. Introduction

Many undergraduate engineering programs require senior students to take capstone design courses, where students have the opportunity to work on capstone projects with realistic requirements that are typically proposed by industry sponsors [1]–[6]. Most, if not all, capstone projects are team-based, requiring multiple students from the same or closely-related disciplines to work as a team [7]–[9]. This helps students gain essential teamwork skills and experience that can launch them into successful future careers [10].

In the Department of Computer Science and Software Engineering at Penn State Behrend, our capstone program spans two semesters, requiring senior students to form teams to design and implement systems to meet realistic requirements originated from industrial customers [11]. This process starts with call-for-proposals sent to potential industry sponsors. The collected project proposals are then exposed to students for them to bid, and student teams are formed afterward.

As shown in Figure 1, a student-centered collaboration model has been implemented. Throughout the two-semester long process, a student team need to work closely with people playing other roles:

(1) Capstone coordinator: The coordinator reviews all the submitted project proposals and selects a set of candidate projects for further consideration. A proposal can be filtered out if it demands knowledge and skills that are far beyond the capability of our students, or it does not impose a reasonable level of difficulties and challenges. Upon receiving bids from students, the coordinator will then form student teams. Each student team consists of three or four students, typically with mixed majors. Once a student team is formed, their project is also fixed. Project assignment is by no means an easy task [12]–[14]. Factors considered by the coordinator include not only student interests, but also whether they may complement each other in terms of skills and personality characteristics.
For each student team, the coordinator will contact the project sponsor to have an industry mentor appointed. The coordinator also helps each student team to form an evaluation committee consisting of one faculty advisor and two co-advisors.

(2) Capstone instructor: The capstone instructor oversees all capstone projects to ensure that engineering principles and standards are consistently applied. The instructor gives lectures on design principles and practices, including code of ethics, software process models, Unified Modeling Language (UML), design patterns, version control, and automated testing. Along the development process, the instructor also collects feedbacks and comments from faculty advisors and industry mentors, which are used to evaluate the performance of each team and every individual students.

(3) Faculty advisor: In order to closely monitor students' progress on their capstone projects and offer guidance in a timely manner, faculty advisors are requested to meet with student teams weekly.
To meet the technical writing requirement, each student team needs to submit project reports twice per semester. The advisor will review the submitted project report and provide evaluation and comments for improvement. In addition, each student team needs to give presentations twice per semester to report their project progress to their evaluation committee (consisting of the faculty advisor and co-advisors). Each evaluator will provide evaluation and comments immediately after a presentation.

(4) Industry mentor: An industry mentor, serving as a real customer of the system under development, plays a sig-

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

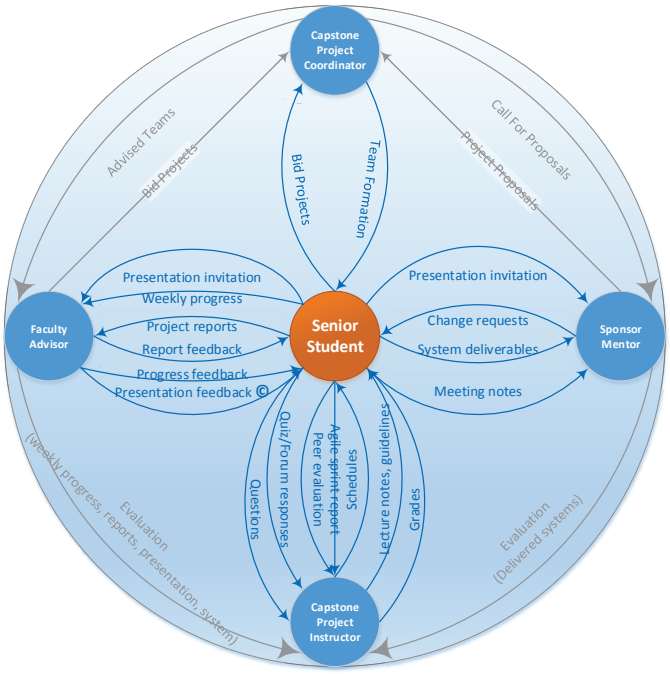

Fig. 1. Student-centered role-based collaboration model.

nificant role in the success of the capstone project [15]–[18]. In our practice, an industry mentor can contribute in three ways: meeting with the student team to clarify user needs and system constraints, reviewing intermediate/final system deliverables, and sending change requests for improvement, if applicable.

Obviously, the success of a capstone project largely relies on the intentional integration of multi-party collaboration.

To foster a better cross-learning environment for students when they work on software-intensive projects, we have adopted the agile SCRUM approach [19], [20] in the development of senior design projects. In agile development, a software system is developed as a series of increments where in each iteration (or sprint) a small piece of working feature is tested and added to the software. A sprint review is conducted at the end of each sprint where the development team demonstrates the working software to the users (including the industry mentor who plays the product owner role and the faculty advisor who serves as the scrum master). After the users have observed or experienced the real value of the software being developed, they can make better decisions about the software's future, which helps the developing team to continuously adapt its plans in subsequent sprints so as to maximize the value it delivers.

Unlike the industrial settings, capstone design students can be challenged when conducting agile sprint reviews. On the one hand, it is extremely hard to bring all the stakeholders together to hear their comments all in one meeting. On the other hand, the availability of industry mentors in particular is mostly unpredictable due to their own busy schedules. To overcome these difficulties, in our practice we have employed two strategies. First, we have implemented a so-called agile buddy approach which allows each team to acquire an outsider student's opinion on their system. Second, we have adopted a web-based collaboration platform such that students can collect critiques asynchronously from different stakeholders.

The rest of this paper is organized as follows. In Section II, we describe the agile sprint activities. In Section III, we introduce the agile buddy approach. In Section IV, we report how change requests are managed and tracked, and Section V concludes the paper.

II. Background on Agile Software Development

The Agile software development approach advocates adaptive planning, early delivery, and continual improvement. To minimize the amount of up-front planning and design, it breaks the development process into short time frames (a.k.a. iterations or sprints), each of which typically lasts from one to four weeks. In each sprint, the development team work closely in all software activities including planning, analysis, design, coding, and testing, with the goal to demonstrate a working product to the stakeholders at the end of the sprint. It favors an incremental development strategy in the sense that each successive release of the product is usable, and each builds upon the previous one by adding user-visible functionality.

As documented in the literature, the Agile approach has been increasingly adopted in capstone courses for computer engineering, electrical engineering, as well as software engineering programs [20]–[23]. In the Department of Computer Science and Software Engineering at Penn State Behrend, we have instructed our students to adopt the agile approach in the development of their capstone projects since 2015. Within each sprint (about two weeks), each student team perform the development activities as shown in Figure 2.

1) Planning & Modeling: In agile development, a team should neither underestimate the amount of planning, nor spend too much time on planning. In the early stage of a project, it is unrealistic to elicit a complete list of system requirements and then make overly detailed plans. Instead, planning spans throughout the agile development process and occurs in each sprint[1]. The development team can start with those requirements (called use cases or user stories in agile terms) which they have confidence, and keep refining existing requirements and discovering new ones for subsequent iterations/sprints.
After the existing requirements are prioritized, the development team can collaboratively make a decision on the scope of work by setting a subset of the requirements as the sprint goals. The team then move forward to design and document the key design artifacts as pertinent to the sprint goals. Lack of overall product design has been

[1]This differs from the waterfall process where extensive project requirements are collected first, followed by long design and coding phases.

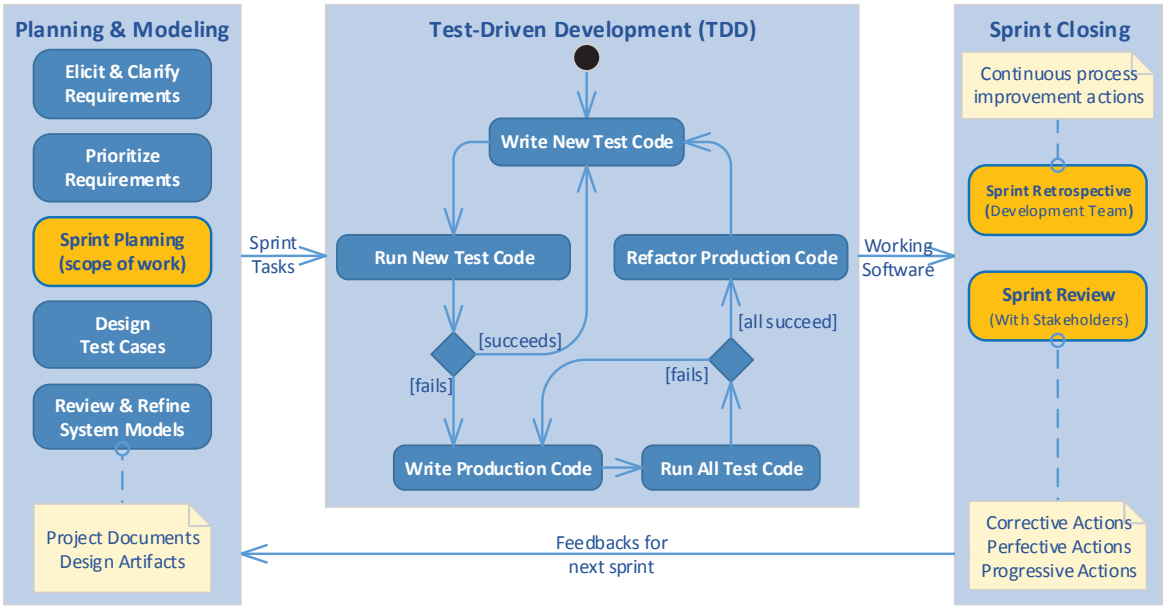

Fig. 2. Activities in a sprint.

identified as a common agile pitfall in the literature. It is true that the focus in agile development is more on producing working software and less on documentation. However, although it is not expected to produce hundreds of pages of documentation as in the waterfall approach, a well-run Agile project should produce the right amount of documentation in each sprint that may include high-level analysis and design artifacts. In our practice, students are instructed to document their design artifacts in the form of use cases, UML class and sequence diagrams. The tangible design document can serve as a roadmap in the sprint, helping the team to identify tasks and trace responsibilities and issues.

2) Test-Driven Development: In TDD a developer constantly repeats the steps of adding test cases that fail, passing them, and refactoring. This has been claimed to have many benefits [24]–[27], some of which are summarized in Table I. In our capstone program, students are instructed to adopt TDD to write test code first for any new features to be developed.
3) Sprint End Activities: A sprint review and a sprint retrospective are conducted at the end of each sprint.
   a) Sprint review: The development team demonstrate the working software to the project stakeholders (including the industry mentor and faculty advisor) and collect their feedbacks about the system under development. The stakeholders label each provided feedback as one of three action types: corrective action, perfective action, and progressive action. The collection of feedbacks acquired in the sprint review provides valuable input to subsequent Sprint Planning.
   b) Sprint retrospective: This is typically done immediately after the sprint review, where the development team deliberately reflect on how they are doing and to find ways to improve their agile activities in subsequent sprints. In particular, each team member is asked to identify specific things that the team should continue doing, stop doing, and start doing.

TABLE I
TEST-DRIVEN DEVELOPMENT OFFERS MANY BENEFITS

	Benefits
1	Lead to a deeper and earlier understanding of the product requirements (ambiguities have to be resolved in the process of transforming a requirements into specific test cases)
2	Ensure the effectiveness of the test code (each test case fails initially)
3	Build a greater level of confidence in the code (all production code is covered by at least one test)
4	Lead to cleaner, well-thought interfaces (in writing a test case, one has to imagine how the functionality is used by clients – the test cases in the first case)
5	Lead to smaller, more focused classes, and looser coupling (product is built as integration of small units that can be written and tested independently)
6	Maintain a continual focus on software quality (refactoring in each cycle of TDD)
7	Avoid lengthy and tedious debugging later in the project (eliminating most defects early in the process)

Agile methods in software engineering have gained popularity and become a standard industry practice in software development. While the agile approach offers great promise, it also presents challenges especially when it is adopted in a college curriculum [28]. Doing agile on campus can be very

much different from an industrial setting. For instance, agile development requires a team to focus only on the committed project. However, in addition to the capstone projects, students on campus often have to take on other course works and activities. Indeed, each student may be able to work only a couple of hours every week on his/her capstone project [29], [30], which is 'anti-agile', making it impossible for a team to conduct some agile practices such as daily standup.

Moreover, customer commitment and customer collaboration are key aspects of project success in agile development [31]. Lack of customer or sponsor support can seldom be an issue in a software development organization. However, for a student team working on their capstone project, the availability of their industry mentor is mostly unpredictable due to his/her own business commitments and schedules. Consequently, it is extremely hard for a student team to bring all the stakeholders together in one meeting to conduct sprint review and hear their feedbacks. In case that their industry mentor cannot partake in the sprint review meeting, the team would face the challenge of missing critical input from a customer perspective. Without timely feedbacks, the team could run into difficulties in planning for subsequent sprints.

To overcome these difficulties, in our practice we have employed two strategies. First, we have implemented a so-called agile buddy approach which allows each team to acquire an outsider student's opinion on their work-in-progress. Second, we have adopted a web-based collaboration platform such that students can collect critiques asynchronously from different stakeholders.

III. Agile Buddy Approach

Our capstone program spans two semesters, and the agile development process starts in mid September and ends in mid April of the next year. This process is further split into 10 agile sprints: one inception sprint and 9 construction sprints. The inception sprint (Sprint 0) starts in mid September and ends in mid October. During this sprint, each student team need to work with their industry mentor to elicit an *initial set* of user requirements, which are then transformed into system requirements. The goal is to have some high-priority system features and constraints identified in the early stage of development.

Each construction sprint lasts for about 2 weeks. A sprint review is conducted at the end of each construction sprint, where a team is asked to demonstrate their working system to both their faculty advisor and industry mentor. The goal is to gather feedbacks from the project stakeholders, refine existing user requirements, and identify new requirements.

Obviously, an industry mentor, serving as a real customer of the system under development, plays a significant role in the success of the capstone project [15]–[18].

According to our experience, most industry mentors are very supportive in attending sprint review meetings. However, it has to be acknowledged that mentors from the industry sponsors are fully engaged in their own business. Some teams did have the experience of not being able to talk to their mentors for two or more sprints in a row.

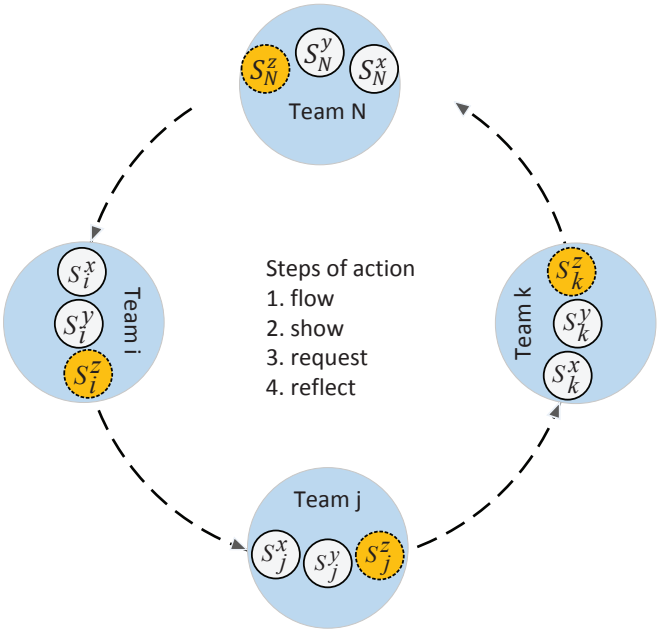

Fig. 3. The Agile Buddy Approach.

Table II shows some example review feedbacks that the two teams collected from their respective industry mentors. For Class2017-Team010, the mentor was able to provide review feedbacks in almost every sprint. However, for Class2018-Team012, the mentor provided feedbacks for only 3 out of 10 sprints. Without timely feedbacks from their mentor, a team could run into difficulties in planning for subsequent sprints and start to worry about their performance and progress.

To encourage cross-team learning and to help student teams collect sufficient user feedbacks, we have coined a term called "agile buddy" as a complementary approach in conducting agile sprint review.

A. Agile Buddy Concept

The agile buddy approach is illustrated in Figure 3. Each team has an agile-buddy role (or ambassador), which can be played by the same student throughout the whole agile process, or played by a different student in different agile sprints. In Figure 3, S_i^z, S_j^z, and S_k^z are used to denote the agile-buddy from Team i, Team j, and Team k, respectively.

The process involves four activities: flow, show, request and reflect.

(1) **Flow:** At the end of a sprint, each agile buddy flows to join another team. For instance, S_i^z joins Team j, S_j^z joins Team k, etc.

(2) **Show:** Each team show the working features of their system under development to the guest agile buddy. For instance, Team j show their work-in-progress to the guest S_i^z; Team k show their work-in-progress to the guest

TABLE II
EXAMPLE SPRINT REVIEW FEEDBACKS FROM INDUSTRY MENTORS

Team ID	*Sprint*	*Industry Mentor*	*Change Type*	*Request Date*	*Detail*	*Priority*
Class2017-Team010	S-01	Anthony Alford	Perfective	10/31/2016	Iterate over github response instead of creating single string	Low
	S-01	Anthony Alford	Progressive	10/31/2016	Begin work on web app framework to combine JS and PY work	Highest
	S-02	Anthony Alford	Progressive	11/14/2016	Next step - connect to a database	Highest
	S-03	Anthony Alford	Progressive	12/13/2016	Good work on setting up the plumbing. Need to define actual schema	Highest
	S-04	Anthony Alford	Progressive	01/24/2017	Will send some AWS credit codes; they may be expired but if not feel free to use if you want to host a mysql instance	Highest
	S-06	Anthony Alford	Perfective	02/21/2017	Connect existing functionality together	Highest
	S-07	Anthony Alford	Perfective	03/21/2017	Looking good, just need to keep adding features!	Highest
	S-08	Anthony Alford	Progressive	04/04/2017	UI looks good, maybe a few more "UX" type improvements	Highest
	S-08	Anthony Alford	Perfective	04/04/2017	Machine learning progressing well. Will look for some ideas of different feature vector values	Highest
	S-09	Anthony Alford	Corrective	04/18/2017	Fix bugs so the system works on other github projects	Highest
Class2018-Team012	S-03	Eric Komorek	Progressive	11/15/2017	The team has developed a great prototype of a panel that overlays on top of a webpage. I am working on getting them access to ERIE's styling library to assist their front-end development.	High
	S-03	Eric Komorek	Perfective	12/11/2017	The students demonstrated a working prototype of a note taking service during their last meeting. Some improvements that we discussed are mainly UI related, as it is in need of finesse and polish.	High
	S-06	Eric Komorek	Progressive	02/08/2018	Discussed the separation of responsibilities, which at this point is Katie: Backend, Karlene: Database and PM, and Brad: UI/Front End.	High
	S-08	Eric Komorek	Perfective	03/19/2018	Need to further polish UI. Remove pagination in favor of scrolling, while implementing searching and filtering functionality.	Highest

S_j^z, etc. During the demonstration, students may freely exchange their practices.

(3) **Request:** After demonstration, each agile buddy sends one or more change requests immediately to the team she/he has just visited. For instance, S_i^z sends change requests to Team j, S_j^z sends change requests to Team k, etc.

(4) **Reflect:** Each student team reflect on the newly collected change requests and discuss how to handle them.

This agile buddy approach not only allows each team to acquire an outsider student's opinion on their system. Furthermore, each team have a chance to collect useful information from other teams. Take Team j as an example, the ambassador S_j^z can bring back Team k's progress and practices, and Team j can also get to know Team i's progress and practices from the guest S_i^z. This not only helps the whole class disseminate best practices, but also can create a bit peer pressure which may push some teams to speed up when they realize that they might have lagged behind other student teams.

B. Agile Buddy Workshop

In practice, the agile buddy concept can be implemented differently.

(1) **Random Reviewing:** Each agile buddy randomly decides which team to review. The downside of this approach is that it can be hard to manage the classroom review activity, and the number of reviews collected by teams can vary greatly.

(2) **Pair Reviewing:** Two teams form a long-term reviewing relationship: the agile buddy from Team A reviews Team B and vice versa. In this approach it's easy to manage the classroom review activity, and the number of reviews collected by teams are balanced. The downside is that, knowing they are reviewing each other, the reviews can be less critical or useful.

(3) **Cycle Reviewing:** All the student teams form a review cycle. In particular, the agile buddy from Team 1 reviews Team 2, the agile buddy from Team 2 reviews Team 3, and so on, and the agile buddy from the last team reviews Team 1. On the one hand, the collected reviews can be very useful and objective because each team reviews a different team and is reviewed by another different team. On the other hand, because each team works on a different project, with a long-term reviewing relationship, it's easier for a team to form a shared mental model with the guest agile buddy about their project. Such a mental model can be incrementally updated as more reviews being conducted over time, which allows the guest agile buddy to easily contextualize the new features of the project under review, thus provide more constructive feedbacks.

The agile buddy approach has been systematically implemented in our capstone program since 2015. A 30-minute agile buddy workshop is scheduled in a computer lab at the end of each sprint. Due to its benefits as explained above, cycle reviewing has been adopted in agile buddy workshops.

Some statistics of sprint reviews for the senior class of 2018 are provided in Table III. Each review is classified as one of three types: Corrective, Perfective, and Progressive. A corrective feedback concerns about detected bugs or defects that should be fixed; a Perfective feedback describes where could be changed to offer a better system performance or user experience; a Progressive feedback suggests to the team what new features may be developed next. Each cell in Table III lists the number of reviews belonging in each of the three types, respectively.

TABLE III
STATISTICS OF SPRINT REVIEWS FOR SENIOR CLASS OF 2018 (CORRECTIVE/PERFECTIVE/PROGRESSIVE)

Team ID	*From Industry Mentor*	*From Faculty Advisor*	*From Agile Buddy*	*Total*
Class2018-Team001	1/0/6	5/0/0	2/5/2	8/5/8
Class2018-Team002	0/1/1	1/0/1	0/1/5	1/2/7
Class2018-Team003	0/0/0	3/0/4	1/3/2	4/3/6
Class2018-Team004	0/0/0	0/1/2	1/3/2	1/4/4
Class2018-Team005	0/0/0	5/0/3	1/2/3	6/2/6
Class2018-Team006	0/0/1	1/0/3	0/3/3	1/3/7
Class2018-Team007	2/2/5	8/5/1	5/2/2	15/9/8
Class2018-Team008	0/0/0	0/0/0	2/3/4	2/3/4
Class2018-Team009	0/1/3	1/0/0	0/1/4	1/2/7
Class2018-Team010	0/0/0	0/0/0	1/2/3	1/2/3
Class2018-Team011	0/0/0	0/0/0	1/2/4	1/2/4
Class2018-Team012	0/3/2	1/3/6	0/3/8	1/9/16
Class2018-Team013	0/0/0	0/0/2	1/1/6	1/1/8
Class2018-Team014	0/0/0	3/1/1	0/0/4	3/1/5
Class2018-Team015	0/0/0	1/0/1	1/2/4	2/2/5
All Teams	3/7/18	29/10/24	16/33/56	48/50/98

We have the following observations: (a) industry mentors rarely provided formal records of reviews (although they did meet with students); (b) the sprint reviews from agile buddies can largely complement those provided by faculty advisors; (c) industry mentrs are more likely to suggest progressive actions; and (d) overall, the number of progressive actions is twice as many as the number of corrective actions or perfective actions.

IV. TRACKING REVIEWS & DISCUSSION

Some sprint reviews can be addressed in the immediate next sprint, some may not. As the development process goes on, students can easily lose track of the status of change requests. In order to facilitate the monitoring of implementation of change requests, CapStone – a web-based collaboration platform – has been utilized. CapStone not only allows students to collect critiques asynchronously from different stakeholders, it also allows the stakeholders to closely monitor students' progress on the implementation of a change request.

CapStone offers different views to different stakeholders:

- A faculty advisor, industry mentor, or agile buddy can only monitor those requests he/she has submitted;
- A student can monitor requests targeted at his/her team from all other stakeholders (including faculty advisor, industry mentor, and agile buddy);
- The course instructor, who oversees the quality of all the capstone projects, can monitor requests targeted at each student team from every stakeholder.

Figure 4 displays a student's view of the feedbacks (change requests) submitted by the faculty advisor, industry mentor, and agile buddies along the agile development process. The change requests are maintained in CapStone as formal records. A change request is in the *Pending* state upon submitted. In CapStone, when a student creates a task in the beginning of a sprint, the task can be linked to one or more change requests, implying that the goal of the task is to address the linked change requests. The state of a change request is changed from *Pending* to *In progress* whenever it is linked to a sprint task in CapStone. A student can check off a change request when the request has been completely addressed; in so doing, its state is changed from *In progress* to *Implemented*.

There are a few interesting questions emerged from our practice of the agile buddy approach. The first question is how to guarantee the commitment of the "outsider" in providing relevant opinion on the other team's work-in-progress. When the first time this approach was adopted, the instructor had planned to assign credits to students' informative reviews to other teams, but it was proved to be unnecessary after just a few sprints because students have not only passionately engaged themselves in the cross-reviewing process, but also demonstrated their professional skills of providing candid critics. As a few examples, the reviews provided by student agile buddies shown in Figure 4 are clear and constructive. When asked, many students said that they were keen to know other team's progress and the agile buddy approach offers such an opportunity. Moreover, quite a few students viewed it as their duty to uphold a high quality in providing reviews to others because they benefited so much from others as well.

The second question is how to distribute the review tasks to all students in a team so that the professional skills acquired by reviewing could be gained by all students. As introduced in Section III-A, which team member takes the role of the ambassador is loosely defined. In other words, it could be always the same team member who plays the ambassador role, but they could also rotate to play this role. The fixed-role approach certainly helps the student to form a better shared mental model over time with the team being reviewed, allowing him/her to continuously provide consistent, constructive feedbacks. The rotating-role approach obviously can address the abovementioned issue of helping all students to develop professional review skills. In our practice, we do not specifically require students to use one approach or the other; they are encouraged to try both instead. However, it would be interesting to quantitatively compare the pros and cons of these two alternatives, which is left as a future study.

Lastly, as a routine, a survey is typically scheduled at the end of the development process where each student has the

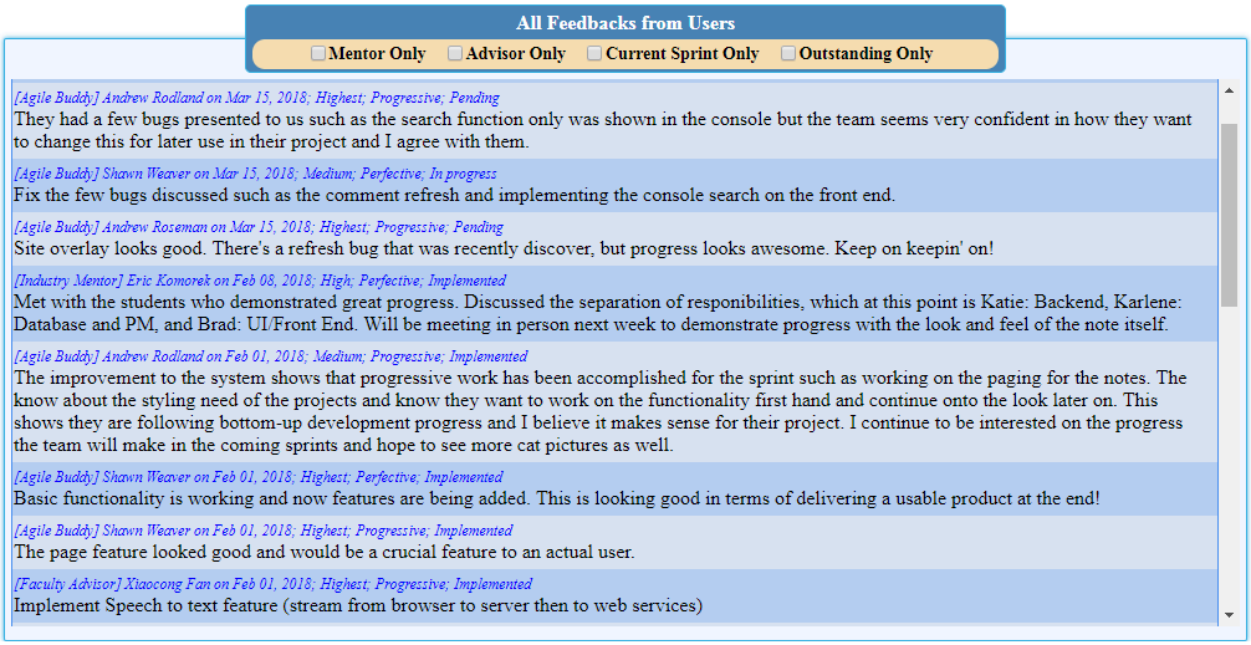

Fig. 4. Sample Feedbacks Collected in Agile Sprint Reviews.

opportunity to reflect on the lessons learned, skills gained, and practices liked or disliked. Among others, the agile buddy approach was listed each year as one of the top three practices that should be continued for the next class of students. It is not without good reason that students liked the agile approach. For example, a team claimed that, if the agile buddy workshops were not used, they would not have considered to utilize the IBM Watson services adopted by another team. Another team changed their server-side technology from Python to Node.JS after chatting with some students in an agile buddy workshop. In short, with all the empirical evidence, the agile buddy approach have been proved to be effective in disseminating best practices among students.

V. Conclusion

In the Department of Computer Science and Software Engineering at Penn State Behrend, a student-centered collaboration model has been implemented in our senior design program. The agile development approach has been practiced along the two-semester long process, where a student team need to work closely with other stakeholders.

Customer commitment and customer collaboration are key aspects of project success in agile development. Lack of customer or sponsor support can seldom be an issue in a software development organization. However, doing agile on campus can be very much different from an industrial setting. For a student team working on their capstone project, it is extremely hard for them to bring all the stakeholders together in one meeting to conduct sprint reviews. In case that their industry mentor cannot partake in the sprint review meeting, the team could run into difficulties in planning for subsequent sprints.

We have employed two strategies to enhance students' experience of agile development. First, we have implemented a so-called agile buddy approach which allows each team to acquire an outsider student's opinion on their work-in-progress. Second, we have adopted a web-based collaboration platform such that students can collect critiques asynchronously from different stakeholders. Our practices in the past three years have confirmed the usefulness of cycle reviewing. On the one hand, the collected reviews are very useful and objective because each team reviews a different team and is reviewed by an agile buddy from another different team. On the other hand, with a long-term reviewing relationship, it's easier for a team to form a shared mental model with the guest agile buddy about their project. Such a mental model can be incrementally updated as more reviews being conducted over time, which allows the guest agile buddy to easily contextualize the new features of the project under review, thus be able to provide more constructive feedbacks.

References

[1] J. L. Ray, "Industry-academic partnerships for successful capstone projects," in *Frontiers in Education, 2003. FIE 2003 33rd Annual*, vol. 3, Nov 2003, pp. S2B–24–9 vol.3.

[2] R. M. Ford and W. C. Lasher, "Processes for ensuring quality capstone design projects," in *Frontiers in Education, 2004. FIE 2004. 34th Annual*, Oct 2004, pp. S2G–12–17 Vol. 3.

[3] D. Ingalsbe and J. Godbey, "Project-oriented capstone course: Integrating curriculum assessment utilizing industry partner and student input," in *Proceedings of the ASEE Annual Conference and Exposition*, Austin, 2005.

[4] J. V. Tocco and D. D. Carpenter, "Re-engineering the captsone: Melding an industry oriented framework and the BOK2," in *Proceedings of the ASEE Annual Conference and Exposition*, Vancouver, BC, Canada, 2011.

[5] R. K. Stanfill and A. Rigby, "The professional guide: A resource for preparing capstone design students to function effectively on industry-

sponsored project teams," in *Proceedings of the ASEE Annual Conference and Exhibition*, Indianapolis, 2014, p. 22.

[6] C. Steinlicht and B. G. Garry, "Capstone project challenges: How industry sponsored projects offer new learning experiences," in *Proceedings of the ASEE Annual Conference and Exposition*, Indianapolis, 2014, p. 11.

[7] P. M. Griffin, S. O. Griffin, and D. C. Llewellyn, "The impact of group size and project duration on capstone design," *Journal of Engineering Education*, pp. 185–193, 2004.

[8] K. F. Li, A. Zielinski, and F. Gebali, "Capstone team design projects in engineering curriculum: Content and management," in *Teaching, Assessment and Learning for Engineering (TALE), 2012 IEEE International Conference on*, Aug 2012, pp. T1C–1–T1C–6.

[9] J. W. Jones and M. Mezo, "Capstone = team teaching + team learning + industry," in *Proceedings of the ASEE Annual Conference and Exposition*, Indianapolis, 2014, p. 7.

[10] C. W. Ferguson and P. A. Sanger, "Facilitating student professional readiness through industry sponsored senior capstone projects," in *Proceedings of the ASEE Annual Conference and Exposition*, Vancouver, BC, Canada, 2011.

[11] X. Fan, "CapStone: A cloud-based platform for multi-party collaboration on capstone projects," in *2016 IEEE Frontiers in Education Conference (FIE)*, 2016, pp. 1–9.

[12] Y. C. Lan and J. A. Ginige, "Towards criteria based allocation of capstone projects for an enhanced learning experience," in *Computer Science and Software Engineering, 2008 International Conference on*, vol. 5, Dec 2008, pp. 121–124.

[13] L. Lopez, M. Aronson, and G. Carstensen, "Optimizing support for senior design project assignments," *Interfaces*, vol. 38, no. 6, pp. 448–464, 2008.

[14] J. E. Burge and G. C. Gannod, "Dimensions for categorizing capstone projects," in *Software Engineering Education and Training, 2009. CSEET '09. 22nd Conference on*, Feb 2009, pp. 166–173.

[15] D. Knudson and A. Radermacher, "Updating cs capstone projects to incorporate new agile methodologies used in industry," in *Software Engineering Education and Training (CSEE T), 2011 24th IEEE-CS Conference on*, May 2011, pp. 444–448.

[16] V. Isomttnen and T. Krkkinen, "The value of a real customer in a capstone project," in *Software Engineering Education and Training, 2008. CSEET '08. IEEE 21st Conference on*, April 2008, pp. 85–92.

[17] J. Vanhanen, T. O. A. Lehtinen, and C. Lassenius, "Teaching real-world software engineering through a capstone project course with industrial customers," in *Software Engineering Education based on Real-World Experiences (EduRex), 2012 First International Workshop on*, June 2012, pp. 29–32.

[18] A. Rusu and M. Swenson, "An industry-academia team-teaching case study for software engineering capstone courses," in *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual*, Oct 2008, pp. F4C–18–F4C–23.

[19] D. A. Umphress, T. D. Hendrix, and J. H. Cross, "Software process in the classroom: the capstone project experience," *IEEE Software*, vol. 19, no. 5, pp. 78–81, Sep 2002.

[20] D. Rover, C. Ullerich, R. Scheel, J. Wegter, and C. Whipple, "Advantages of agile methodologies for software and product development in a capstone design project," in *Frontiers in Education Conference (FIE), 2014 IEEE*, Oct 2014, pp. 1–9.

[21] J. C. R. Stansbury, M. Towhidnejad and M. Dop, "Agile methodologies for hardware/software teams for a capstone design course: lessons learned," in *Proc. ASEE Annual Conference*, 2011.

[22] V. Mahnic, "A capstone course on agile software development using scrum," *IEEE Transactions on Education*, vol. 55, no. 1, pp. 99–106, Feb 2012.

[23] M. Grimheden, "Increasing student responsibility in design projects with agile methods," in *Proc. ASEE Annual Conference*, 2013.

[24] K. Beck, *Test-Driven Development by Example*. Boston, MA: Addison Wesley, 2002.

[25] M. F. Aniche and M. A. Gerosa, "Most common mistakes in test-driven development practice: Results from an online survey with developers," in *2010 Third International Conference on Software Testing, Verification, and Validation Workshops*, 2010, pp. 469–478.

[26] Y. Rafique and V. B. Misic, "The effects of test-driven development on external quality and productivity: A meta-analysis," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 835–856, 2013.

[27] W. Bissi, A. G. S. S. Neto, and M. C. F. P. Emer, "The effects of test driven development on internal quality, external quality and productivity: A systematic review," *Information and Software Technology*, vol. 74, pp. 45 – 54, 2016.

[28] A. Martin, C. Anslow, and D. Johnson, "Teaching agile methods to software engineering professionals: 10 years, 1000 release plans," in *Agile Processes in Software Engineering and Extreme Programming*, H. Baumeister, H. Lichter, and M. Riebisch, Eds. Cham: Springer International Publishing, 2017, pp. 151–166.

[29] M. Kropp and A. Meier, "Teaching agile software development at university level: Values, management, and craftsmanship," in *2013 26th International Conference on Software Engineering Education and Training (CSEE T)*, 2013, pp. 179–188.

[30] D. Kumar and I. Govender, "Bringing agile practice to the classroom: Student voices of third-year major project implementation," *African Journal of Science, Technology, Innovation and Development*, vol. 8, no. 5-6, pp. 423–428, 2016.

[31] S. C. Misra, V. Kumar, and U. Kumar, "Identifying some important success factors in adopting agile software development practices," *Journal of Systems and Software*, vol. 82, no. 11, pp. 1869 – 1890, 2009.