Developing Communities of Practice to Prepare Software Engineers with Effective Team Skills

Ann Q. Gates¹, Elsa Y. Villa², and Salamah Salamah¹
¹Department of Computer Science and ²Center for Research in Engineering and Technology Education
The University of Texas at El Paso, U.S.A.

ABSTRACT

A major challenge to teaching software engineering is achieving functioning teams that enforce individual accountability while integrating software engineering principles, approaches, and techniques. The two-semester, software engineering course at the University of Texas at El Paso, referred to as the Team-Oriented Software Engineering (TOSE) course, establishes communities of practice that are cultivated through cooperative group practices and an improvement process model that enables learning from past experiences. The experience of working with incomplete, ambiguous and changing software requirements motivates the need for applying disciplined software engineering practices and approaches throughout project development. Over the course of the two-semester sequence, the nature of students’ participation in project teams changes, they begin to influence others in software engineering practice, and their identities as software engineers begins to develop. The purpose of the chapter is to describe how to structure a software engineering course that results in establishing communities of practice in which learners become increasingly more knowledgeable team members who embody the skills needed to work effectively in a team- and project-based environment.

INTRODUCTION

A long-standing problem when teaching software engineering is achieving functioning teams that enforce individual accountability. Working as teams, students complete a large project while going through the appropriate training in team skills. The human aspect of software development also makes teaching software engineering and managing student-run projects challenging because of the following:

- general lack of maturity in the students’ team and communication skills,
- difficulty in ensuring that all team members contribute to the project,
- differences in students’ experiences and understanding, and
- difficulty in evaluating and ensuring individual and team progress and work quality.

A two-semester, software engineering course, referred to as the Team-Oriented Software Engineering (TOSE) course, at the University of Texas at El Paso (UTEP) addresses these challenges by incorporating cooperative-learning principles with an aim of establishing a community of practice. Cooperative learning as an instructional approach (Johnson, Johnson, & Holubec, 1992; Johnson, Johnson, & Smith, 1991) is an evidence-based practice that contributes to team building while increasing student
achievement and self-esteem (Johnson & Johnson, 1989). Using cooperative learning principles to structure groups generates positive interdependence in which each member is committed to supporting others in reaching their goals while at the same time working together to meet the group goal. The emphasis on cooperative behavior cultivates an environment in the software engineering course where communities of practice can emerge and grow. Drawing from the work of Lave and Wenger (1991) and Wenger (1998), a community of practice is defined as a group of individuals who share a common purpose, contribute to each other’s success, and develop shared practices that identifies them as members of that group.

The purpose of the chapter is to describe how structuring a software engineering course using cooperative learning principles results in establishing communities of practice in which learners become increasingly more knowledgeable team members who embody the skills needed to work effectively in a team- and project-based environment. The objectives of the chapter are to: (1) present the challenges in developing functional teams; (2) outline how to structure a software engineering course in which teams move toward becoming a community of practice; and (3) describe how a community of practice serves to support functioning and practicing software engineers.

BACKGROUND

Introduction

In the perspective of this chapter, cooperative learning is at the core of building functional and effective teams for addressing the issues, challenges, and concerns of ineffective student teams typically resulting from ill-structured group work (rather than team work). In such group work, a task is given to the group with the hope, for example, that group members will resolve any conflicts on their own and allow for a “leader” to emerge who can take charge. When groups are structured in this manner, those who are “followers” minimally contribute to deliverables and may be marginalized by the others. Rather, a cooperative learning framework is one that is intentionally designed to promote positive social interaction among the group members through the use of techniques and tools, such as active listening, active participation, conflict management, and individual accountability. When purposefully structured in a sustained manner, team members eventually embody these interpersonal skills; the functionality of the team reflects one of cooperation where team members have mutual respect and trust; and members’ contributions are valued. In this case, the cooperative team most resembles an authentic community of practice. Using the term “authentic” distinguishes it from what is now a ubiquitous way to describe communities in action as “communities of practice” when too often they may simply be groups of individuals working together with a common purpose. In this section, more detailed descriptions of cooperative learning and communities of practice are discussed.

Cooperative Learning

Human beings are fundamentally cooperative, yet most individuals will argue that humans are competitive. This myth prevails in spite of the often-used phrases of only the strongest survive and survival of the fittest. Interest in cooperation as a scholarly endeavor began in early twentieth century with the work of social psychologist Kurt Lewin (see Lewin, 1935; 1948) who investigated group dynamics and interplay of its members. Morton Deutsch, a student of Lewin’s, continued investigating group dynamics and extended it to formulate a theory of competition and cooperation (see Deutsch, 1949; 1962). By the mid-20th century, David Johnson, under the guidance of Deutsch, further investigated competition and cooperation, including perspective taking and conflict, during the 1970s (Johnson, 1974; 1975). These studies were further extended to identify the following outcomes of cooperative learning: higher achievement, increased retention, greater intrinsic motivation, increased perspective taking, more positive heterogeneous relationships, higher self-esteem, and greater collaborative skills (Johnson &
Further efforts resulted in the identification of five essential components of cooperative learning (Johnson, Johnson, & Holubec, 1990; Johnson, Johnson & Smith, 1998) described in the next section.

The five essential elements of cooperative learning in the Johnson and Johnson model are as follows:

1. **Positive Interdependence**: Students are linked to others in the team in such a way that they cannot succeed unless their teammates do and that they must coordinate their efforts with the efforts of other members to complete a task. It is important to set clear team goals and tasks so that members believe that they cannot succeed unless everyone succeeds.

2. **Promotive Interaction**: Members of the team promote each other’s success by sharing needed resources and helping, supporting, encouraging, and praising each other’s efforts. Promotive interaction is characterized by individuals providing each other with efficient and effective help and assistance, as well as providing each other with feedback in order to improve their subsequent performance.

3. **Individual Accountability**: The group must be accountable for achieving its goals. Each member must be accountable for contributing his or her share of the work. The group has to be clear about its goals and be able to measure its progress in achieving them and the individual efforts of each of its members. Individual accountability exists when the performance of each individual student is assessed, the results given back to the individuals and the group, and the student is held responsible by team members for contributing his or her fair share to the group’s success.

4. **Professional and interpersonal skills**: Group members must know how to provide effective leadership, decision-making, trust-building, communication, and conflict management and be motivated to use these skills. Skills must be taught and practiced.

5. **Group processing**: Group processing refers to reflecting on a team activity to describe which team behaviors were helpful and not helpful and to make decisions about what actions to continue or change. Team members describe how well they are achieving their goals and maintaining effective working relationships, as well as identify members’ actions that are productive or not. The reflection helps clarify the team’s goals and effective working relationships. Indeed, continuous improvement results from careful analysis of how members are working together and determining how team effectiveness can be enhanced.

The model requires the instructor to structure activities incorporating these five elements. Later sections describe how this is done in the TOSE course.

**Theoretical Underpinnings.** As mentioned earlier, the scholarly work of David and Roger Johnson builds on the work of Lewin and Deutsch in investigating how social interaction supports learning. They posit that social interaction occurs in three forms: competitive, cooperative, and individual. In describing the effect of cooperation in comparison to competition, Johnson and Johnson (1989) describe this as the result of bidirectional relationships:

Deutsch’s (1985) crude law of social relations states that the characteristic processes and effects elicited by a given type of social interdependence also tends to elicit that type of social interdependence. Thus, cooperation tends to induce and be induced by mutual help and assistance, exchange of needed resources, influence, and trust. People tend to trust their collaborators but also to seek out opportunities to collaborate with those they trust. There is a benign spiral of cooperation in which cooperation promotes trust, trust promotes greater cooperation, which promotes greater trust, and so forth. (p. 9)
Also contributing to this theory are other strands of psychology, such as social, cognitive, and behaviorist. Most notably among these is the contribution of social psychologist Lev Vygotsky (1978, 1986) who theorized learning happening in social contexts through the interplay among and between individuals, objects, and language. Thus, learning is cultural and occurs in interaction with others. Other contributions include educator and philosopher John Dewey who wrote extensively about the importance of social interaction with others.

[In] this intercommunication one learns much from others. They tell of their experiences and of the experiences which, in turn, have been told them. In so far as one is interested or concerned in these communications, their matter becomes a part of one’s own experience. Active connections with others are such an intimate and vital part of our own concerns (Dewey, 2009, p. 9).

**Communities of Practice**

Coined by Lave and Wenger (1991), the term *communities of practice* describes a concept of how novices, such as apprentices, in a particular discipline or craft learn the work of those more expert. As Wenger (1998) noted, communities of practices draw from scholarly work in sociology, anthropology, and psychology. He further posited that, as one participates in the practices of a community, meaning is negotiated in engagement with others, ideas, language, and objects. This draws in part from Vygotsky’s (1978, 1986) socio-cultural theory of learning whereby individuals acquire concepts, theories, and the discourse of the group as their participation in the community gradually increases.

[Learning then results from] practices that reflect both our pursuit of our enterprises and the attendant social relations. These practices are thus the property of a kind of community created over time by the sustained pursuit of a shared enterprise (Wenger, 1998, p. 45).

The practice of sharing expertise and resources with others in a community allows members to thrive in appropriating such practices. Communities of practice are formed over time as participants engage in activities with a common purpose that defines the existence of the group. Newcomers enter communities either by invitation, as is the case for established communities, or by situation, as is the case for a college classroom where access is open to those who qualify in terms of their academic qualifications. In this case, however, the group is merely a collection of individuals who have the propensity to evolve into a community of practice when the group reflects the following characteristics:

1. Sustained mutual relationships
2. Substantial overlap in participants’ descriptions of who belongs
3. Knowing what others know, what they can do, and how they contribute to the enterprise
4. Displaying certain styles of membership
5. Mutually defining identities
6. Shared ways of engaging in doing things together
7. Quick setup of problem to be discussed
8. Shared discourse
9. Ability to assess appropriateness of actions and products
10. Specific tools, representations, and other artifacts
11. Local lore, shared stories, inside jokes
12. Jargon and other communication shortcuts

Using the categorization defined by Wenger (2013), the aforementioned characteristics could be classified as follows: *the domain*-the group has an identity defined by a shared domain of interest and goes beyond
being a connection among members (characteristics 1-5); the community-members interact and learn from each other (characteristics 6-9); and the practice-members are practitioners and develop a repertoire of resources (characteristics 10-13).

Theoretical Underpinnings. Learning theories have historically been theorized by cognitive psychologists and scientists, and more recently by social scientists as the product of our social and cultural interaction with others in thought and language. One of the preeminent theories to recently emerge is situated learning theory (Lave & Wenger, 1991), i.e., we learn values, language, knowledge and skills situated in everyday practice with others. Learning is situated and happens when learners engage with others in authentic and meaningful (i.e., real world) practice. Having a common purpose motivates participation of group members. Wenger (1998) postulated that communities of practice evolve where groups of people gather for a common purpose, such as family, church, or formal classrooms. In this interaction, we interpret what is said and what is done to obtain meaning of a practice. Members of the community gradually adopt these practices through immersion in the practice with others in the community. In communities of practice, learning the practices of the group takes time as newcomers, or novices, learn from those more expert as they socially interact in the practices of the community.

TEAM SKILLS DEVELOPMENT: ISSUES AND PROBLEMS

A recent paper (Radermacher & Walia, 2013) presented the results of a systematic literature review identifying the areas of greatest divergence between skill sets required by the industry and the actual skills of graduates. Teamwork, oral communication, project communication, and problem solving were identified as the skills with the widest gaps. Surveying engineering competencies of over 4000 graduates in eleven different engineering disciplines, Passow’s study (2012) solicited opinion about the relative value of Accreditation Board for Engineering and Technology (ABET) prescribed student outcome areas. The ability to function on a team and written and oral communication skills were among the highest rated competencies.

Peter Denning (1992) further noted the lack of skills in recent graduates as follows:

Employers and business executives complain that graduates lack practical competence. Graduates, they say, cannot build useful systems, formulate or defend a proposal, write memos, draft a simple project budget, prepare an agenda for a meeting, work on teams, or bounce back from adversity; graduates lack a passion for learning. They say the current concepts-oriented curriculum is well suited for preparing research engineers, but not the practice-oriented engineer on which their competitive advantage increasingly depends. (p. 5)

Although educators have made much progress in preparing graduates to meet the competencies as described by Denning, a significant gap still exists between computer science graduates’ capabilities and the needs/expectations of the industry. Because of the importance of team, communication, and other professional skills in the future workforce, a number of universities offer dedicated courses for engineers to develop professional skills. Furthermore, many computer science and software engineering programs include professional skills development in multiple courses using a variety of techniques and exercises to teach, for example, communication and team skills (Coleman & Lang 2012). Communication skills are typically taught through instructor lectures and practiced through student presentations and written reports (Liu, Sandell & Welch, 2005).

Students often complain that, although they are expected to work in teams on projects, they are seldom given any advice or guidance on how to work in a team (Hart & Stone, 2002). Or, if they are given guidance, it may often from a business perspective that students find difficult to integrate into their software development practice. Techniques for building team skills include lectures on group process and
diversity. Team skills can be built through engagement in activities aimed at requiring individuals to complete a given task critical to the success of the team project (Lingard & Berry 2002).

In software engineering capstone courses, students are typically grouped into teams to work on a one- or two-semester development project. Such courses provide the ideal setting for teaching and enhancing professional skills since they provide an opportunity to mimic real-world industrial settings (Broman, Sandahl, & Baker, 2012; Hogan & Thomas, 2005) through the use of team projects to apply the technical skills acquired through the coursework. One feature of these team projects is the need for communicating ideas to other members of the team, yet explicit directions or exercises to help students enhance these skills are often lacking (Smith, 2000). As mentioned in the Background section, merely grouping students without specific structure is a risk in achieving a desired outcome, particularly for developing teaming and communication skills. Swan, Magleby, Sorensen, & Todd (1994) noted that being a member of a dysfunctional team may result in students developing negative views of the whole experience of teamwork. Finally, McGinnes (1995) emphasizes that skills development, such as teaming, communication, and project management, should be integrated throughout the undergraduate curriculum rather than isolated in capstone courses.

THE TOSE COURSE: SOLUTIONS AND RECOMMENDATIONS

Drawing on both cooperative learning and communities of practice concepts, this section addresses the aforementioned challenges of software engineering projects by offering guidelines on how to incorporate cooperative learning into teams to eventually move them toward exhibiting the attributes of a community of practice. The structure for the Team-Oriented Software Engineering (TOSE) course uses a two-prong strategy: (1) develop student teams using a cooperative learning framework and (2) cultivate an environment to immerse teams in the practice of software engineering. TOSE is modeled after the computing research teams where cooperative learning principles were integrated into group meetings with an aim of developing students’ skills as researchers. Findings from a qualitative study by authors Gates and Villa (Villa, Kephart, Gates, Thiry, & Hug, 2013) revealed that these research teams had attributes resembling those of an authentic community of practice as previously defined.
The project deliverables associated with the first semester (requirements engineering) of a TOSE course include the following: feasibility study, interview report, prototype, and software requirements specification document. The use case diagrams, data flow diagram, class diagram, and state transition chart are integrated into the documents. The deliverables for the second semester (design and implementation) are as follows: architectural design, detailed software design document, software code, and test plan. As the project teams work on the deliverables, they evolve into communities of practice in which team members become increasingly more knowledgeable and effective software engineering practitioners. The cooperative approach prepares teams that function effectively and encourages continuous improvement of teams together with the software development process. Fig. 1 elucidates the synergy between cooperative learning elements and communities of practice.

In particular, the TOSE course provides a foundation for project teams to evolve into communities of practice in which team members become increasingly more knowledgeable software engineering practitioners. To produce effective and functioning software engineering teams, instructors must first invest their time in understanding and then skillfully applying cooperative learning principles, in particular the deliberate and intentional professional skills building, in facilitating the software engineering teams. Like any good design, extensive planning is needed to facilitate an environment where students are intrinsically motivated and engaged in their learning.

The following section is organized around the elements of a community of practice, as defined by Wenger (1998, 2013): domain, community, and practice. For each element, we describe how cooperative learning is structured in the course and provide example activities of the TOSE course.

Figure 1: The TOSE strategy.
Setting the Domain

A community of practice sets the *domain* or identity of shared interest (Wenger, 1998). While typical communities of practice are created around an affinity and expertise for a particular topic, a software engineering course brings together a diverse group of students, who may or may not know each other and who typically have various levels of skills and capabilities. Rather than allowing students to self-select into their teams, the TOSE course emulates an actual work environment—students submit résumés featuring their professional and educational experiences, including relevant coursework. They apply for positions in which they are interested in taking the lead role, rank the positions in order of preference, and justify their choices. The positions are requirements analyst, system designer, detailed designer, lead programmer, and verification and validation supervisor. The lead for each position is responsible for the quality of the assigned deliverable and accomplishes this by delegating work, integrating individual assignments, interfacing with course instructors, and ensuring that deliverables are submitted on schedule. The instructor evaluates each application and determines the membership of the project teams based on each student’s expertise, grade point average, choice, and personality. Students self-identify their personality based on a simple personality exercise.

Establishing positive interdependence among the team members lays the foundation for setting the domain for the team to evolve into community of practice. Positive interdependence can be structured in the following ways: create positive role interdependence (each member is assigned a role), resource interdependence (members rely on a common resource to accomplish a task), positive reward interdependence (celebrate team’s successes), goal interdependence (mutually shared team goal), and identity interdependence (establish a mutual identity).

After students are assigned to teams early in the semester, the team members brainstorm on a name for their “company” and create a logo, thus, establishing *identity interdependence*. *Goal interdependence* is inherent in the course goal since students work throughout the course to complete the course milestones and deliverables by the assigned due dates. Assigning each member of the team with a specific job during phases of the project establishes *role interdependence*. Roles are also assigned on in-class assignments, and leaders are asked to assign roles during meetings. Examples of roles include recorder, summarizer, direction giver, time manager, devil’s advocate, and questioner. Through role assignments, students learn behaviors that are essential for teamwork. Resources, which could include such things as project materials, specialized equipment, or tools, are limited; thus, team members must determine how best to share these resources, which is an example of *resource interdependence*. Finally, because the students’ final grade includes how well they function in their team, students coordinate their efforts to complete tasks and realize that their success is dependent on the success of the other team members.

As a way of modeling promotive interaction, the instructor can acknowledge each team member as an expert in a skill they identified in their résumé. In addition, integrating role assignment throughout the course encourages students to help, support, encourage, and acknowledge each other’s efforts.

Building Community

A community of practice engages in joint activities and discussions, helps one another, and shares information (Wenger, 2013). In addition to the two-semester TOSE project, the instructor assigns readings, delivers a brief lecture to reiterate important concepts from the readings, engages the class in discussion, and seeks clarification as needed. In-class assignments are used to build expertise in software engineering techniques, approaches, and tools. Students work in small groups to complete the assignment, which are structured as cooperative groups in which members share information and help each other learn.

For example, consider an assignment in which students are asked to create a state transition diagram for a given problem. Students can work in teams to create a diagram. The instructor can assign the role of
devil’s advocate to a student where he or she must challenge another member’s contribution and reasoning. Such an exercise may lead to higher quality decision-making and can provide greater insight into problem solving. In an extension of this assignment, the instructor could use one or more diagrams to have the students conduct an inspection. For such an activity, the instructor would create a group of four or five students and assign the roles of leader, scribe, and inspectors. To help the students learn how to run an inspection, the instructor could prepare a script for the inspection, which would include a checklist that the inspectors would use to determine if the authors are using correct notations, and the script could have the inspectors review particular aspects of the diagram to determine if the notations capture the intended meaning of the problem and if the model is complete. The scribe would be instructed to record the responses and findings. A grade would be assigned based on the effectiveness of the inspection in finding shortcomings.

An essential TOSE component that leads to building community is constructive critique of assignments. Time is allocated during the class session to define, apply, and reflect on constructive critique and its importance as a team skill. Students are expected to resubmit work based on instructor and peer feedback to improve their understanding and add to their expertise. The instructor schedules weekly half-hour meetings with the teams to provide constructive critique of work-in-progress, ensure that the team is on track with respect to their next deliverable, and to ensure individual accountability. In addition, the instructor encourages team leaders and members to acknowledge contributions of others and praise each other’s efforts, an example of the cooperative learning element of promotive interaction. It is important for the professor to emphasize constructive critique and to model promotive interaction while interacting with the groups. In a sense, the teaching assistant and instructor are the experts, and the students are the apprentices. It is important to advocate for their individual effort in achieving mutual goals, achieving the team goal, and becoming motivated to strive for mutual benefit. Practicing promotive interaction and incorporating positive interdependence are important in a developing a community practice.

**Becoming a Software Engineering Practitioner**

The TOSE course aims to move students from novices to practitioners by the end of the second semester; however, becoming a practitioner develops over time due to the numerous skills and software engineering knowledge students need to more fully participate in a community of software engineering practitioners. This section describes the TOSE course structure and the activities for practicing cooperative team, professional, and communication skills.

As described earlier, to move students toward becoming software engineering practitioners, the TOSE course requires students to apply for one of five positions within each team. The instructor provides the teams with a requirements definition document that is the starting point for working with a customer to elicit requirements for a project that is too complex for any one person to complete. As a result, students must work together throughout the year to develop the project from inception to implementation. When the teams are formed, the student, who has been assigned the lead for a particular deliverable, will distribute the tasks needed to complete the deliverable. The instructor provides activities in which teams learn software engineering concepts needed to complete the deliverable. In addition, the teams practice team skills and reflect on how to improve the essential elements of functioning teams, including holding students individually accountable for their contributions to deliverables.

Earlier sections describe how positive interdependence and promotive interaction are structured in the project and in-class teams. To structure individual accountability, another element of cooperative learning, the instructor must assess individual efforts in mastering software engineering knowledge. There are several avenues for assessing students’ team skills: observation, reflection, and weekly meetings with the instructor or assistant as described earlier. Observation forms (Scholtes, 1995; Johnson, Johnson & Smith, 1998) are used by the instructor to record particular behaviors observed during a team meeting and provide the feedback to the team at the end of the meeting. For example, the
instructor may mark how often team members seek input from other members and how a team member ensures that the discussions are on task. The feedback provides the team with information on how they interact and identifies areas of improvement.

In a community of practice where members share resources, it is important to query members about his or her contributions, observe the team to check the frequency with which each member contributes, and have members explain concepts to others. In a TOSE course, the team leader completes a Task Assignment Record in which he or she logs the date and task assigned to each team member, and the date that the assignment was submitted. Together with meeting records, correspondence, and memoranda or e-mails, this paper trail is maintained by the team leader in a team notebook, thus, providing a reliable means of determining individual contributions.

Another effective approach is to have each team turn in a document at the end of the semester that summarizes individual, subgroup, and group contributions; each member of the team is required to sign the document to acknowledge their agreement with its contents. If a member does not agree and cannot come to consensus with the group after discussion, then a meeting is called with the professor to discuss differences. It is imperative that, at the beginning of the semester, students understand that they will be held individually accountable for producing quality work towards the team's deliverable.

For students to learn and be motivated to apply effective leadership, decision-making, communication, and conflict management skills, the instructor must integrate activities where these skills are taught and practiced. First, the TOSE instructor needs to assess the teams’ abilities and identify appropriate skills to be learned and practiced. Second, the instructor must design an in-class activity, as previously described, and assign roles to students related to that skill to serve as practice. Prior to practicing skills, the instructor involves the students in describing the verbal and visual cues of a person applying the skill. Finally, after the group activity, the instructor asks students to reflect on how well they were able to apply the skill, and provides opportunities for them to continue their practice of the skills until they become automatic. As an example, consider teaching the skill “asking a probing question,” which prepares students for the interview with the customer. To teach this, the instructor would ask the class for the verbal cues that are indicators that the question is probing. The class might contribute the following responses: the question does not have a yes or no answer; the question seeks to extend knowledge; the question would begin with the words “how” and “what;” and the question would require a comparison or analysis. The visual cues are not as appropriate for this particular skill, but could include raised hand, eye contact with the speaker, and nodding head (to indicate listening and seeking understanding). Students could be asked to create a list of probing questions over a particular topic and the questions could be critiqued with feedback to the author.

Basic skills, e.g., active participation checker, active listening checker, and recorder, can be taught early in the first semester and practiced by assigning particular roles during in-class group discussions. The skill of active participation checker is one in which a person ensures that all group members participate by calling on them or asking them to share their opinion of what others have contributed. The instructor identifies other skills from feedback by team members of issues or concerns arising from their team meetings. For example, if a team provides feedback that members come away from a meeting with different impressions about their newly assigned tasks, the instructor can teach the role of summarizer or paraphraser and describe the importance of assigning someone this role.

The instructor provides students guidelines for conducting productive meetings and a template for planning an agenda and documenting the meeting. The template includes a section for entering the attendees’ names, assigning roles, entering past action items for status checking, and setting the agenda with time allotment. The leader is responsible for ensuring that the following occurs during the meeting:

- the discussion remains focused on the topic and moves along,
- no one dominates or is overlooked,
• discussions come to a close, and
• the group is notified when the time allotted for an agenda item has expired or is about to expire.

Each meeting should have a scribe who records the names of those in attendance, key subjects and main points raised, decisions made including action items, and items that the group has agreed to defer or raise again at a future meeting. Team members can refer to minutes to reconstruct discussions, find decisions made, and review assigned action items. This role is rotated among the team members to ensure all learn these skills.

Every team meeting should include actions to facilitate discussion. It is expected that team leaders use skills for effective discussion; however, the team will be even more successful if each team member learns and practices them. While the following techniques are discussed in the framework of team meetings, they are useful whenever effective discussion is important. The behavior and roles that are important for conducting an effective meeting, which are learned over time, are as follows:

• Ask for clarification when the topic being discussed or the logic in another person’s arguments is unclear (Role: clarifier/paraphraser).
• Encourage equal participation among group members by directly asking a member to express his or her opinion or making a general request for input, especially in the case when a member dominates the conversation (Role: active participation checker).
• Actively explore a person’s ideas rather than debating or defending each idea that comes up (Role: elaborator).
• Occasionally compile what has been said and restate it to the group in summary form followed by a check for agreement (Role: summarizer).
• Ensure that there are no overlong examples or irrelevant discussion (Role: time keeper).
• Manage time by reminding the team of deadlines and time allotments so work can be either accelerated or postponed, or time readjusted appropriately (Role: time keeper).
• Be prepared to end a discussion when there is nothing to be gained from further discussion and help close the discussion (Role: facilitator).
• Summarize the group’s position on an issue, state the decision that seems to have been made, and check whether the team agrees with the summary (Role: integrator).

Finally, group processing, another cooperative learning element, provides an opportunity for a team to improve how it functions. To ensure that group processing takes place, instructors should allocate some time for team members to process how effectively members worked together. In industry, this is referred to as an “after-action report” or “post-mortem review.” While after-action reports take place at the culmination of a project, group processing in the TOSE course occurs throughout the two semesters as a mean of providing continuous quality improvement. To ensure effective group processing, sufficient time must be allowed for sharing; a structure must be provided for processing (such as “list three things your group is doing well today and one thing you could improve”); positive feedback is emphasized; the processing is specific rather than general; students are actively involved; instructors remind students to use their cooperative skills while they process; and the instructor communicates clear expectations about the purpose of processing.

After a TOSE course major deliverable, the instructor asks students to write answers to the following basic questions: What are your contributions? What is working? What is not working? What can be improved? While the first question targets individual accountability, the remaining questions address group functioning. Students are told to write comments using first person as a way of capturing the individual’s perception of the group. In addition, students are asked to phrase criticisms using a constructive tone. After reviewing the responses, the professor or teaching assistant groups them by teams, strips the headers, deletes the contribution section, and edits the text if personal remarks have been
The altered responses along with comments from the instructor are sent to the respective teams. The teams discuss the content and devise strategies for team improvement. It is imperative that the instructor meets with groups who are not functioning well. The processing identifies teams with problems allowing for early intervention. The approach aligns with improvement process models such as Plan-Do-Check-Act (Deming, 1986) that provides a feedback cycle to enable learning from past results.

In addition to practicing cooperative skills, students review through class discussion the following stages of team growth (Tuckman, 1965), as well as the roles that group members can practice to assist in developing their ability to function well in the team. Roles prescribe what an individual is obliged to do and provides a mechanism for building positive interdependence and developing interpersonal skills.

**Stage 1: Forming.** In this stage, members explore the boundaries of acceptable group behavior and test the leader’s guidance. This stage includes the feelings of excitement, optimism, tentative attachment to the team, suspicion, fear, and anxiety. It also includes the following behaviors: complaints about the organization and barriers to the task, discussion of problems not relevant to the task, impatience with discussions, and decisions on what information needs to be gathered. The roles that support how a group functions are as follows:

- **Explainer of Ideas or Procedures** - shares one's ideas and opinions.
- **Recorder** - writes down the group's decisions and edits the group's report.
- **Gatekeeper** - ensures that all members are contributing and gives both verbal and nonverbal support and acceptance through seeking and praising others' ideas and conclusions.
- **Direction Giver** - gives direction to the group's work by reviewing the instructions and restating the purpose of the assignment, calling attention to the time limits, and offering procedures on how to complete the assignment most effectively.
- **Clarifier /Paraphraser** - restates what other members have said to understand or clarify a message.

**Stage 2: Storming.** The Storming stage is the most difficult: Team members realize that the task is different and more difficult than realized; members become testy, blameful, or overzealous. This includes feelings such as resistance to task and quality improvement approaches, and sharp fluctuations in attitude about team and project’s chance of success; it also includes behaviors such as arguing among members even when they agree on real issues, defensiveness and competition, questioning of wisdom of guidance team, leaders, establishing unrealistic goals, concern about excessive work, and some disunity, tension and jealousy. Roles that can help a team ferment ideas and resolve conflict are as follows:

- **Criticizer of Ideas, Not People** - intellectually challenges member by criticizing their ideas while communicating respect for them as individuals.
- **Asker for Justification** - asks members to give the facts and reasoning that justify their conclusions and answers.
- **Differentiator** - differentiates the ideas and reasoning of group members so that everyone understands the differences in members’ conclusions and reasoning.
- **Integrator** - integrates the ideas and reasoning of group members into a single position to which everyone can agree.

**Stage 3: Norming.** Members in the Norming stage reconcile competing loyalties and responsibilities; they accept the team and the individuality of team members. This stage includes feelings such as a new ability to express criticism constructively, acceptance of membership in the team, and relief that everything is going to work out. It includes behaviors such as an attempt to achieve harmony by avoiding conflict, more friendliness, confiding in each other and sharing of personal problems; team cohesion, common spirit and goals, establishing and maintaining team ground rules and boundaries (norms). As team members get used to working together, their initial resistance fades away. Some roles that can help formulate information are as follows.
• Summarizer- restates the group’s major conclusions or answers or what has been read or discussed as completely and accurately as possible without referring to notes or to the original material.
• Accuracy Coach- corrects any mistakes in another member’s explanations or summaries and adding important information that was left out.
• Perspective-taking roles- each member is responsible for contributing one’s perspective or viewpoint to the group’s final product.

Stage 4: Performing. The team has settled its relationships and expectations. They can begin performing, diagnosing and solving problems, and choosing and implementing changes. All members have discovered and accepted each other’s strengths and weaknesses, and learned what their roles are. It includes feelings such as members having insights into personal and group processes, and better understanding of each other’s strengths and weaknesses. It includes behaviors such as constructive self-change, ability to prevent or work through group problems and close attachment to team. As team members become more comfortable with each other, and better understand the project and what is expected of them, they become a more effective unit with everyone working in concert. The following roles can move a team toward higher levels of thinking:
• Extender- extends the ideas and conclusions of other members by adding further information or implications.
• Prober- asks in-depth questions that lead to analysis or deeper understanding.
• Cognitive roles- each member is responsible for contributing one aspect of the critical-thinking process to the group’s final product (e.g., analysis, synthesis, evaluation, elaboration, application).

Understanding these stages of teaming allows student teams to reflect on the characteristics they may be experiencing. By providing roles for them to assume, the teams are more likely to transition from forming to performing teams.

FUTURE RESEARCH DIRECTIONS

As aforementioned, results from a qualitative study by authors Gates and Villa (Villa, Kephart, Gates, Thiry, & Hug, 2013), which focused on alumni who were former members of research groups using TOSE instructional approaches, provides evidence of graduates’ ability to effectively transfer communication and team skills when they enter the workforce. A future research effort is to conduct a comparative study of TOSE and non-TOSE graduates to assess their level of preparedness in effectively working in teams upon entry into and their advancement in the workforce over time. Social media tools are an emerging technology; such tools are currently not a feature of TOSE. Thus, an interesting future research effort would be to investigate the effectiveness of these tools in supporting team communication, individual accountability, and group processing. Finally, another research direction would be to individually follow students through TOSE using ethnographic research methods. This would contribute to a better understanding of specific features of TOSE that contribute (or hinder) development of identities as software engineers.

CONCLUSION

The paper describes the TOSE two-prong strategy: (1) develop student teams using a cooperative learning framework and (2) cultivate an environment to immerse teams in the practice of software engineering. As a result, project teams evolve into communities of practice in which team members become knowledgeable and effective software engineering practitioners. The cooperative approach provides a framework for teams to learn what is needed to function effectively and to incorporate continuous quality improvement into the software development process.
The two-semester TOSE course incorporates the deliberate practice of team skills while providing students with an introduction to approaches, techniques, and methods to structured and object-oriented software engineering methodologies. Students work in teams to develop a cross-disciplinary, large-scale software system. The course provides student teams with an opportunity to deal with the challenges of developing a real-world product. The experience of working with incomplete, ambiguous and changing requirements motivates the need for applying disciplined software engineering practices and approaches, as well as effective team work throughout project development.

Students are assessed on their individual knowledge of software engineering through exams and homework assignments, which constitute 55 percent of their grade, and they are assessed on their team project, which constitutes 45 percent of their grade. The team receives a team grade on each deliverable and a grade for the team and individual presentation at the end of the semester. The team grade may vary among members based on the team notebook that documents individual contributions and group contributions to deliverables; leadership abilities on deliverables; and ability to describe and apply cooperative team skills. The chapter describes several avenues for assessing team skills and providing the team with feedback that enables their ability to improve how they work in teams: observation, reflection, and weekly meetings with the instructor or assistant.

In every phase of software development from elicitation to modeling and from analysis to design and implementation, students are instructed in techniques and principles of software engineering, then asked to apply this knowledge to their project. For example, shortly after the discussion of techniques for elicitation of requirements, the client is brought to campus for an interview. Immediately after discussing team development, strategies for conducting effective meetings, and time management, students are assigned to project teams and asked to keep minutes of their meetings and task assignments, which are handed in and graded along with other project deliverables. During the project lifetime, the project guidance team (consisting of faculty and teaching assistants for the course) guide project teams through the software development process.

To produce effective and functioning software engineering teams, instructors must commit to investing their time in understanding and then applying cooperative learning principles, in particular the deliberate and intentional professional skills learning, in facilitating the software engineering teams. Once instructors gain an understanding of how to skillfully apply cooperative learning principles into their software engineering course, they must reflect on its effective use in supporting community building and adjust as needed. They must also build in features for students to actively reflect on how these communities develop into professional software engineers in order for them to build such teams when they join the workforce. Like any good design, extensive planning is needed to facilitate an environment where students are intrinsically motivated and engaged in their learning.

REFERENCES


**ADDITIONAL READING SECTION**


Hazzan, O., & Har-Shai, G. (2013). Teaching computer science soft skills as soft concepts. *Proceedings of the 2013 Special Interest Group on Computer Science Education Technical Symposium*


KEY TERMS AND DEFINITIONS

Cooperative Learning: A teaching approach in which students work in groups and formally structured to include the following elements: positive interdependence, promotive interdependence, skills developments, individual accountability, and group processing.

Communities of Practice: A group of individuals who mutually engage in pursuing a common purpose through shared resources and repertoire of concepts, stories, discourse, and action.

Positive Interdependence: Team members share a common purpose with each contributing to the success of others and to the group goal(s).
Promotive Interaction: Team members interact in positive ways by acknowledging, for example, the contributions of others.

Situated Learning: Negotiated meaning of values, language, knowledge and skills in the practice of doing.

Professional Skills: A set of interpersonal skills to energize and influence a team in meeting its goal(s), such as oral and written communication skills, team skills, and leadership skills.

Team-Oriented Software Engineering (TOSE): A software engineering course using cooperative learning principles as a pedagogical approach to building effective teams and cultivating an environment with an aim of developing a community of practice.