

Chapter 73

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

Ann Q. Gates

The University of Texas – El Paso, USA

Elsa Y. Villa

The University of Texas – El Paso, USA

Salamah Salamah

The University of Texas – El Paso, USA

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.

DOI: 10.4018/978-1-5225-3923-0.ch073

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 identify 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

Overview

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

18 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/developing-communities-of-practice-to-prepare-software-engineers-with-effective-team-skills/192946

Related Content

Fault-Tolerant and Fail-Safe Design Based on Reconfiguration

Hana Kubatova and Pavel Kubalik (2011). *Design and Test Technology for Dependable Systems-on-Chip* (pp. 175-194).

www.irma-international.org/chapter/fault-tolerant-fail-safe-design/51401

Some Key Topics to be Considered in Software Process Improvement

Gonzalo Cuevas, Jose A. Calvo-Manzano and Iván García (2018). *Computer Systems and Software Engineering: Concepts, Methodologies, Tools, and Applications* (pp. 134-160).

www.irma-international.org/chapter/some-key-topics-to-be-considered-in-software-process-improvement/192875

Fractal Coding Based Video Compression Using Weighted Finite Automata

Shailesh D. Kamble, Nileshsingh V. Thakur and Preeti R. Bajaj (2021). *Research Anthology on Recent Trends, Tools, and Implications of Computer Programming* (pp. 232-252).

www.irma-international.org/chapter/fractal-coding-based-video-compression-using-weighted-finite-automata/261029

Agile Development Processes and Knowledge Documentation

Eran Rubin and Hillel Rubin (2018). *Computer Systems and Software Engineering: Concepts, Methodologies, Tools, and Applications* (pp. 1433-1453).

www.irma-international.org/chapter/agile-development-processes-and-knowledge-documentation/192930

A Brief Overview of Software Process Models: Benefits, Limitations, and Application in Practice

Sanjay Misra, Martha Omorodion, Luis Fernández-Sanz and Carmen Pages (2018). *Computer Systems and Software Engineering: Concepts, Methodologies, Tools, and Applications* (pp. 1-14).

www.irma-international.org/chapter/a-brief-overview-of-software-process-models/192870