2
Teamwork

Abstract

This chapter presents one of the basic elements of software projects—teamwork. It addresses how to build teams in a way that promotes team members' accountability and responsibility, and that fosters communication between teammates. One of the basic ways to start team building is by assigning roles to the team members. For this purpose a role scheme is presented in this chapter, according to which each team member is in charge of a specific managerial aspect of the development process, such as design and continuous integration, in addition to his or her development tasks. Teamwork is not always a simple process, and sometimes it raises dilemmas and conflicts between team members. This aspect of teamwork is not neglected in agile teams, and when a conflict emerges, it is addressed openly by all the team members. In the section that deals with teamwork in learning environments, it is illustrated how the role scheme and the discussion about dilemmas in teamwork provide an evaluation framework for software projects developed by student teams in academia.

2.1 Overview

This chapter focuses on teams—one of the main and most influential factors of software projects' success. Consequently, it is highly appreciated and supported by agile software development methods.

One practice which is highlighted in this chapter is applying a role scheme, according to which each team member has an additional role in the team in

O. Hazzan, Y. Dubinsky, *Agile Software Engineering*,
DOI: 10.1007/978-1-84800-198-5_2, © Springer-Verlag London Limited 2008

addition to being a software developer. This role scheme fosters the interconnections and dependencies between the members of agile teams and enhances creativity, responsibility, accountability, diversity, and measure collection. Further, this role scheme shows that each team member can contribute to software development on the team level, beyond his or her individual contribution, and that the mutual contributions of the individuals in the team create a whole which is greater than the sum of its parts.

To highlight the importance attributed to teamwork in agile software development environments, this chapter also presents a set of activities that deal with roles, which may form the needed atmosphere for agile teamwork.

In addition, we discuss potential dilemmas in teamwork and how agile development may help cope with such dilemmas. Specifically, we examine how to satisfy the individual needs of each team member and, at the same time, achieve the needed contributions to the team's work.

Based on this examination, an evaluation scheme for student software projects, developed in teams, is presented in the section of this chapter that deals with teamwork in learning environments. This evaluation scheme takes into consideration the various conflicts that may arise.

2.2 Objectives

- Readers will become familiar with the characteristics of software teams in agile software development environments.
- Readers will learn how to allocate roles to members of agile teams and how to exploit the benefits of role assignment at the individual, team, and organization levels.
- Readers will discuss dilemmas in teamwork and understand how agile teams can overcome them.
- Readers will get a sense of how agile team spirit can be achieved, empowered, and maintained.
- Readers will gain basic skills to exploit the strength of agile teams.

2.3 Study Questions

1. Find definitions for the concepts of team and teamwork. Discuss the definitions' relevance for software teams.
2. What is the purpose of teams? Why are teams needed?

3. As a software team member, how would you like to feel in your team? What values would you like to pursue? What practices would you want to see in your team development environment to provide an atmosphere that fits you?
4. What roles do you have in your life? How do you manage to perform them all? Describe two personal scenarios in which a conflict emerged from the need to play different roles and explain how the conflicts were solved.
5. Why should roles be assigned to software team members? What roles would you assign to your team members?
6. Look through the literature for different approaches to role assignments in software teams in general and in agile software teams in particular. Compare these approaches: What ideas do they share? How do they differ from each other? What are the specific characteristics and responsibilities of each role holder in the team?
7. Discuss at least three problems which software teams face. Suggest ways to solve these problems.
8. Suggest dilemmas which software team members might face. Suggest ways to solve these problems. Suggest agile principles and practices that can support such cases.

2.4 A Role Scheme in Agile Teams

According to Humphrey (2000), a team consists of at least two people who are working towards a common goal/objective/mission, in which each person has been assigned a specific role to perform and in which a completion of the mission requires some form of dependency among team members (p. 19). In the case of a software project, a team is a group of individuals who have gathered to produce a software product.

In software projects teams are needed for the accomplishment of the complex task of software development. It is however, not a trivial task to mange software teamwork. This is partially because software development is about an intangible product, one that cannot be seen, smelled, or touched, and therefore, the development status and the exact responsibilities are not always clear.

The unique situation of software development can be approached in different ways. We will start with role assignment, which is one way by which agile software development methods attempt to overcome the typical challenges of software projects. Role assignment means that each team member has an additional role besides that of developer.

The assignment of roles serves as a means for splitting the responsibility for project management and progress among all the team members. The rationale for this stems from the fact that one person (or a small number of developers) cannot manage the entire richness and complexity involved in software development projects. When the responsibility is split among all teammates, each aspect of the process is treated by one teammate, and each teammate feels a personal responsibility for that specific aspect. Both the software project as a whole and each of the individual team members benefit from this kind of organization.

Table 2.1 presents a role scheme for an agile software team, which expands and integrates the role schemes suggested by the different agile methods. It is based on the idea that the responsibility for the software's progress and success should be transferred to and distributed among *all* teammates (Dubinsky and Hazzan 2004, 2006).

There are four groups of roles.

The first is the *leading group,* which consists of the coach, tracker, and methodologist. It is important to note that the tracker and methodologist in this group, as well as the other role holders, do not reduce the significance of the project leader/coach. To the contrary—as it turns out, the role scheme improves the project leader's position and provides him or her a better way to assess and lead the development process.

The second is the *customer group,* which consists of the user evaluator, customer, and acceptance tester. If the project has a real customer, the roles in the customer group should be conceived of as a bridge between the real customer and the team. If the project does not have a real customer, these role holders play a real customer.

The third group is the *code group,* which is composed of four roles: designer, unit tester, continuous integrator, and code reviewer. This group of roles focuses on those aspects of software development that are directly related to the coding activity.

The fourth group is the *maintenance group,* which comprises three roles: presenter, documenter, and installer, and focuses mainly on the product's external presentation and documentation.

As can be seen, the different roles address different aspects of the development process (leadership, customer, code, and maintenance), and together they encompass all the aspects of a standard software development process.

Holding a role, and thus being in charge of a specific aspect of the development process, *does not* mean that the role holder performs all the activities related to his or her particular role; instead, it implies that each role holder must ensure that the specific aspect for which he or she is responsible will be carried out properly by all team members. Since this idea applies to all team members, the project management is split among all team members and, at the same time, is covered by all of them.

Table 2.1 Roles in an agile software team (Reprinted from Journal of System Architecture, 52, Dubinsky Y, Hazzan O. Using a role scheme to derive software project quality, 693–699, Copyright (2006), with permission from Elsevier. Also, with kind permission of Springer Science and Business Media.)

Group of roles	Role	Description
Leading group	Coach	Coordinates and solves group problems, leads and guides development sessions
	Tracker	Measures the group progress by measures as defined by the team, the customer, and the organization; manages the workspace boards; manages the team diary/collective memory. See also Chapter 5, Measures
	Methodologist	Makes sure that the team works according to the defined development process, answers questions related to the methodology, looks for solutions to problems related to the methodology
Customer group	User evaluator	Performs an ongoing user evaluation of the product (collects and processes feedback received from real end users), holds a user centric approach, serves as the user interface designer. See also Chapter 3, Customers and Users
	Customer	If the project doesn't have a real customer: tells customer stories, makes decisions pertaining to each iteration, provides feedback, defines acceptance tests. See also Chapter 3, Customers and Users
	Acceptance tester	Defines (with the customer) and develops acceptance tests, inspires a test-driven development process. See also Chapter 6, Quality
Code group	Designer	Maintains current design, works to simplify design, searches for refactoring tasks and ensures their proper execution. See also Chapter 8, Abstraction
	Unit tester	Establishes an automated test suite, guides and supports others in the development of unit tests, guides a test-driven development process. See also Chapter 6, Quality
	Continuous integrator	Establishes the integration environment; publishes and encourages rules pertaining to the addition of new code, including testing issues
	Code reviewer	Maintains source control, establishes and refines coding standards, guides and manages the team's pair programming
Maintenance group	Presenter	Plans and organizes iteration/release presentations, demos, and roles; measures presentations
	Documenter	Plans and organizes the project documentation: process documentation, user's guide, and installation instructions
	Installer	Plans and ensures the development of an automated installation kit, maintains the collaborative workspace infrastructure

For example, let us assume that one of the teammates is a developer who also has the role of unit tester. As the unit tester, this team member is in charge of all the unit testing activities of the entire project, and of guiding other teammates in the development of their unit tests. But this does not paint the entire picture: let's look at this team member from the perspective of being a developer. As a developer, this team member should write quality unit tests for his or her own development tasks. As the person who is specialized in unit testing, it makes sense that she or he will write quality tests, which can in turn serve as examples for the other teammates of how unit tests should be written. In addition, as a developer, this team member is guided with respect to other aspects of the development process by team members who hold the other roles. This scenario shows how the two hats of each team member—a role holder and a developer—are interconnected and contribute to the development process, improve the software quality, and reinforce the team members' communication.

Task

In a way similar to the analysis presented above with respect to the unit tester role, analyze at least three additional roles from Table 2.1.

The cumulative impact of all the roles increases the team members' commitment to the project. In order to carry out a role successfully, each team member must gain a global view of the developed software and to be involved in all parts of the application, in addition to carrying out his or her personal development tasks. If a team member has a limited view and is aware only of his or her tasks, he or she will not be able to perform the personal role properly. The need to accomplish the personal role satisfactorily actually increases one's involvement and accountability, as well as the commitment to the development process, and leads one to become familiar with all the software parts. Consequently, communication and knowledge sharing, which are vital for software development, are once again increased among team members.

In general, in having a personal role, the team members are expected to perform their development tasks as well as the tasks related to their personal role. Thus, no teammate is merely a developer. The two activities have a mutual positive influence, and consequently the collaboration and communication between the team members is enhanced and the agile team's spirit is maintained. Further, the dual functionality of each team member increases the transparency of both the software process and the product. The process becomes more transparent because it is clear who is in charge of each of its aspects; the product becomes more transparent because each team member is familiar with all the product components and aspects, at least with respect to his or her role.

This perspective is different from the approach in which each role holder is responsible for the entire implementation of the aspect of which he or she is in charge, while the other team members do not perform it at all (e.g., the documenter is the

only team member who is involved in documentation activities). While this approach may lead other team members to reduce their responsibility with respect to that activity, the role scheme described above just inspires the opposite: a strong interconnection exists between the teammates. One team member cannot properly perform all the tasks involved in a software project, even with respect to only one aspect of the development process; cooperation between teammates is essential in order to accomplish the development process properly and on time. Further, since the role scheme clarifies the exact responsibilities of each team member, team members are committed to each other by ensuring that all of them are able to perform their roles in the best way. All these messages delivered by the role scheme increase the team members' involvement and commitment to the project and to the team.

To sum up: Roles are important for the establishment and maintenance of agile teams. A clear role scheme inspires an agile spirit and contributes both to the individuals, to the team, and to the project's success. Further, the role scheme spreads the leadership and management of the software project among all the team members. It lets the developers know that not all the responsibility is carried by one person.

2.4.1 Remarks on the Implementation of the Role Scheme

- The set of roles that a software development method includes in its role scheme reflects the values that a software development method attempts to inspire. Therefore, the role scheme that a software development method defines is one of the key elements of the method. Indeed, different agile methods suggest different role schemes that support their values and conception of software development processes.
- In addition to the role definitions presented in Table 2.1, several roles also support communication between the four groups. For example, the installer is also in charge of communication with the code group.
- At the first stages of the development process, or when the team is established, the role holders should learn their roles and establish a procedure that will enable them to perform their role properly. In the next stages of the agile project, role holders should maintain the spirit and the actual performance of the aspect that their role focuses on.
- When teams consist of fewer than twelve developers, several roles can be unified and assigned to one team member. There are different ways to unify roles, and each has its own advantages. In each case, however, the entire list of roles should be assigned and performed by all team members.

- The team can choose whether each of its members will specialize in one role for a long period of time or, alternatively, whether the roles will rotate among the team members. The exact way by which it is done in practice should be set by each team according to the team members' preferences. For example, it can be decided that roles are reassigned at the beginning of each release.
- Such a team organization eases project management, since it is clear who is in charge of what aspect of the development project, what aspect should be treated by whom, and who should be approached when a specific problem, which belongs to a specific aspect of the development process, arises. Even in cases when there are role overlaps, they will not interrupt the process. Sometimes they can even foster project development. One example is when the unit tester and the acceptance tester work together to introduce test-driven development.

Tasks

1. How does the role scheme reflect the HOT perspectives of agile methods?
2. Predict what attitudes and feelings such a role scheme might raise in agile software teams.
3. What information does each role holder need in order to perform his or her role successfully?
4. In what ways does the role scheme relate to the Agile Manifesto?
5. Describe how each of the Agile Manifesto principles is supported by the role scheme.
6. What benefits does role rotation have?
7. Suggest a mechanism for role rotation in agile teams. What are its benefits? What are its pitfalls?

2.4.2 Human Perspective on the Role Scheme

Social Aspect

- A personal role increases teammates' involvement, communication, accountability, responsibility, and commitment to the software development process and to their team.
- Team members wish to have a specific role in addition to their development tasks in order to increase their influence and involvement in the project management.

Cognitive Aspect

- Since each team member approaches the product from one specific perspective, each can focus on this one specific aspect without being distracted by the multifaceted nature of software product development. In other words, on the global level, the role definition encourages each team member to treat the software product from one perspective. Consequently, each gradually improves his or her understanding about that aspect.
- The role scheme supports the thinking of the development process on multiple levels of abstraction (see Chapter 8, Abstraction). Since abstraction is a key component of software development, every mechanism that supports team members' thinking in terms of different levels of abstraction should be enhanced. On the one hand, each team member sees his or her development task on a relatively low level of abstraction; and on the other hand, the personal role of each team member enables each of them to gain a global overview of the developed system on a higher level of abstraction. Agile methods support thinking at different levels of abstraction in additional ways, such as short releases (see Chapter 3, Customers and Users) and refactoring (see Chapter 8, Abstraction).
- The role scheme enhances knowledge distribution, since each team member specializes in one domain and shares his or her knowledge with the other team members. In addition, since the role scheme leads to knowledge distribution, no harm happens when one team member leaves the team. Indeed, he or she has gained expertise in his or her role; at the same time, however, parts of this knowledge have already been spread. Thus, if a team member leaves the team, the other team members have a reasonable amount of knowledge to continue with respect to that role.
- The role scheme supports the individual's professional development. Team members perform their roles and improve their role performance while learning the practice that their role represents. In turn, they become experts in the specific aspect of software development on which their personal role focuses. In addition, when a team member feels that he or she has exhausted one role's contribution to his or her professional development and wishes to hold another role in the team, as has already been mentioned, role rotation can take place.

Tasks

1. To each of the ideas presented in the human perspective on the role scheme, add its organizational and technological view.
2. Analyze the role scheme from the organizational and the technological perspectives.

2.4.3 Using the Role Scheme to Scale Agile Projects

The role scheme also supports the scaling up of agile projects. Suppose we have five agile teams as part of one software project, and each of them applies the role scheme. In this setting, weekly role meetings are set for each role, in which all the role holders from all the teams participate. For example, a weekly meeting of all testers of the project takes place; a biweekly meeting of all the integrators takes place, etc. It is recommended that these role meetings be scheduled at the same time, in order not to collide with the development sessions of the project teams. In these meetings project-wide issues are discussed, so that the project management proceeds in one direction.

The use of the role scheme for scaling up purposes also enhances knowledge distribution. On the individual level, each team member has the opportunity to communicate with other developers, beyond his or her team, to present the knowledge his or her team has gained so far with respect to a given role, and to serve as a bridge between the team and the organization with respect to that aspect of development of which she or he is in charge. On the team level, each team may benefit also from the wisdom and experience gained by other teams. For example, the team representatives may bring into the role meetings a problem which their team faces, and ask the other role representatives whether their experience can contribute to a solution. Such a dialogue creates a knowledge infrastructure for the development process from which all teams can benefit. On the organization level, and based on the individual and team levels, knowledge is distributed, managed, and maintained.

The role scheme also supports measures related to the project's progress (see Chapter 5, Measures). The measures and policies that should be applied by all the teams enable the project's management to know on an ongoing basis the project's status, progress, and quality. Based on this information, management monitors and controls the project's progress.

2.5 Dilemmas in Teamwork

One of the problems that can arise with respect to teamwork is the question of how to allocate incentives, rewards, and bonuses among team members.

This question is relevant with respect to many professionals and kinds of institutions. However, reward allocation in software engineering is important mainly, but not only, because teamwork is essential in software development. As a result, conflicts between the required cooperation on the one hand, and one's desire to excel as an individual on the other, may intensify. The discussion is

especially relevant with respect to *agile* teams since teamwork is one of the basic working assumptions of agile software development, and team members are asked to cooperate, share information, and exchange ongoing feedback with the other players in the development environment.

Task

This task is based on Hazzan (2003). It aims at elevating the developers' awareness to these potential conflicts and to encourage discussing them openly. This approach is in agreement with the first principle of the Agile Manifesto: individuals and interactions over processes and tools.

Perform the task presented in Figure 2.1 with your team.

Step 1 of the task focuses on the individual's preferences; step 2 examines how team members face possible conflicts between their own preferences and the preferences of the other team members. Thus, in the case of new teams, this activity also fosters the team members' acquaintance with each other.

The discussion that takes place at step 3 focuses on the team preferences at the individual and at the team level. This discussion can be promoted by the following reflective questions.

Step 1: Individual work

You are a member of a software development team. Your team is told that if the project it is working on is successfully completed on time, the team will receive a bonus. Five options for bonus allocation are outlined below. Please explain how each option might influence team cooperation, and select the option you prefer.

	Personal Bonus (% of the total bonus)	Team Bonus (% of the total bonus)	How this option may influence teammates' cooperation
A	100	0	
B	80	20	
C	50	50	
D	20	80	
E	0	100	

Step 2: Teamwork (to be facilitated with the development team)

Each team decides on one option that all team members, as a group, prefer.

Step 3: All teams discussion

Discuss with all the teams the processes that took place in the above two steps.

Figure 2.1 *Bonus allocation activity [© 2003 ACM, Inc.].*

Reflective Questions

1. What were your considerations when choosing your personal option for bonus allocation?
2. Did you face conflicts while working on this task individually (Step 1)? What was their source? How did you overcome these conflicts?
3. Did you face conflicts while working on this task with your team (Step 2)? What was their source? How did you overcome these conflicts?
4. What questions, emotions, and dilemmas with respect to software teams were raised during individual and team work?
5. Predict what considerations would cause developers to prefer a different option for bonus allocation than yours.
6. What characterized the discussion in your team about the agreed upon option for bonus allocation? How did the team agree about the preferred option?

2.6 Teamwork in Learning Environments

The studio meeting this week focuses on activities related to the introduction of the role scheme, the role assignment, and the grading policy that is used for the evaluation of the students' work. The details appear in the continuation of this section.

2.6.1 Teaching and Learning Principles

The following teaching and learning principle deals with the role scheme. (In our list of teaching and learning principles presented in Chapter 14, Delivery and Cyclicality, this is principle number 7.)

Teaching and Learning Principle 7: Assign Roles to Team Members.

According to this principle, each team member has both an individual role, chosen by the member from a given list (for example, coach, unit tester, acceptance tester, code reviewer, etc.), and development tasks for which he or she is responsible.

Such a role scheme does not imply that each role holder carries out all activities related to the domain for which he or she is responsible; rather, each role holder makes sure that the activities related to hir or her domain are accomplished satisfactorily by all team members. Accordingly, the assignment of roles helps divide the responsibility for project progress and management among all team members.

The rationale for this principle is that one person (or a small number of team members) cannot be responsible for the entire richness and complexity involved in software development. When the responsibility is divided among all team members, each aspect of the entire process is addressed by one team member, and at the same time each team member feels personal responsibility for that specific aspect. Both the project itself and the team members benefit from this arrangement. Furthermore, the need to perform one's role successfully actually forces all the team members to be involved in, and to become familiar with, all parts of the developed application. Consequently, knowledge sharing, communication, and involvement are enhanced among team members.

2.6.2 Role Activities

We present the actual application of the role scheme through three kinds of activities. The first kind deals with the role assignments. Second, activities that maintain the role performances on a daily basis are described. Third, an activity that aims at improving the role performances is presented. The activities can be performed in both academic and industrial settings.

2.6.2.1 Role Assignment Activities

The first two activities introduce the role scheme to the team members. If the activities are carried out in an industrial setting, they should be facilitated when the agile team is established, in order to let the team members feel the interconnection among themselves, and their mutual responsibility as an agile software development team. The other activities should be facilitated as development proceeds.

Figure 2.2 describes the first activity related to role assignment. It focuses on the creation of one agreed upon role list.

Since in this studio meeting the academic coach sits together with ten to twelve students who do not know each other but will soon start working together on many tasks related to software development, this activity initiates the students' relationships as teammates.

Time: In academia: second meeting; in industry: when an agile method starts to be implemented.
Task description: You are going to develop a software product as a team. Write down the roles that in your opinion should be performed as part of your project.
Individual work (10 minutes): Students/developers write down their lists.
Team discussion (20 minutes): Students/developers discuss their suggestions, trying to generate one agreed upon list.
For students: Students are asked to prepare a prioritized list of roles they would prefer to perform during the semester.
Summary: The academic coach/agile facilitator presents the role scheme (Table 2.1).

Figure 2.2 *Activity 1: role list generation (Reprinted from Journal of System Architecture, 52, Dubinsky Y, Hazzan O. Using a role scheme to derive software project quality, 693–699, Copyright (2006), with permission from Elsevier.).*

In industry, this activity signals the beginning of a change in the team structure with respect to personal responsibilities. It can be facilitated when the team is first introduced to agile software development, as part of a workshop that the team attends (see Chapter 12, Change) or at the beginning of the implementation phase of agile software development.

In both cases, Activity 1 improves the team members' acquaintance of and familiarity with their teammates.

Activity 2 (Figure 2.3) describes the role distribution.

In academia, the full set of roles is determined for the entire semester. This full implementation is needed for the evaluation process, presented later in this chapter, which is based on the fact that all students have the same load.

Also, according to the role scheme presented in this chapter, the students are responsible for the software's progress and success. Therefore, it is important to note that the academic coach should *not* be the team coach, and that the role of coach should be given to one of the students. The academic coach is in charge of

Time: In academia: second meeting; in industry: following the previous activity.
Discussion (10 minutes): In your opinion, how should we assign roles to teammates?
Task description (20 minutes): Distribute the roles among the teammates. At the end of the task suggest a list of role-teammate pairs.
Discussion (15 minutes): Discuss what portion of your time you should dedicate to the performance of your personal role and what portion should be devoted to the accomplishment of your development tasks?
Reflection (after the second meeting in academia; during a team discussion in industry):
- Express your opinion about the process of role assignment in your team.
- How do you conceive of your role? What input would you expect to get from your teammates with respect to your role?

Figure 2.3 *Activity 2: role distribution (Reprinted from Journal of System Architecture, 52, Dubinsky Y, Hazzan O. Using a role scheme to derive software project quality, 693–699, Copyright (2006), with permission from Elsevier.).*

project evaluation and control from the academic perspective, but does not lead the actual development process. This perspective gives the academic coach a better way to assess the development process and the teammates' work by means of the grading policy, presented later in this section, that supports the role scheme and is based on both an individual component and a team component.

In an industrial setting, the role scheme should be applied in a gradual fashion. It is recommended that the team decide with what roles they would prefer to start the agile implementation phase. The team chooses several roles to start with, and team members volunteer to carry out these roles. It is recommended that the team start at least with the roles of coach, tracker, and unit tester, and gradually add roles according to the team preferences, needs, and adjustment to the agile process.

The actual role assignment itself has a direct influence on team communication. For example, in an industrial setting, it opens new horizons to team members: they are exposed to new facets of their teammates of which they were not aware, even though they may have worked together for many years.

This activity also influences and enhances learning. For example, team members may suggest that roles should be assigned not according to what is appealing, but rather according to what area one is *not* skilled in. In this way, in order to perform their role properly team members will have to communicate about the various aspects of their role with other teammates who are more knowledgeable. Thus the team members will learn new topics.

2.6.2.2 Role Maintenance Activities

The activities in this part are performed on a regular (daily, weekly, iteration) basis. In academia this enables the academic coach to be aware of the project's progress and to improve the students' work assessment; in industry it enables the entire team to be aware of the project status.

Stand-up meeting: Stand-up meetings take place every day in industry, and on a weekly basis in academia. It can be decided that some portion of the brief personal report (one or two sentences) be dedicated to the personal role. Each team member reports about his or her role performance and about his or her expectations from teammates with respect to personal roles.

Presentations to customers: The following task fits for an academic setting; when appropriate, it can be adjusted for an industrial setting. Specifically, each presentation to the customer consists of two parts. In the first part, the development tasks of the iteration are presented; in the second part, each student briefly presents how he or she improves product quality by the accomplishment of his or her role.

The preparation for these presentations takes place one week before the presentation of each iteration to the customer. The students can discuss what the presentation should contain, how much time will be dedicated for each part, and how the personal

roles will be presented. Since students usually do not have previous experience in presenting software products, they will benefit a lot from this activity.

Feedback after presentations: In academia, there are three presentations to the customer during the semester: at the seventh, eleventh, and fourteenth (the last) meetings (see Chapter 1, Introduction to Agile Software Development, for a semester schedule). After each presentation two reflective activities are carried out: the first is a feedback session that takes place with the all team members; the second is a personal reflection that encourages the students to evaluate their work in the last iteration. As part of this reflection, students are requested to evaluate their role performance as well as to grade it.

2.6.2.3 Role Improvement Activity

The following activity can be requested by the academic coach periodically when she or he observes that it is needed. Such a need occurs mainly in cases when the academic coach feels that some roles are not being performed properly. The students are asked to summarize their role activities, to publish their summaries in the electronic forum, and to provide feedback to the summaries presented by the other team members.

2.6.3 Student Evaluation

One outcome of the team discussions that take place in the bonus allocation activity (Figure 2.1) is an agreement that each individual's contribution should be considered on the basis of his or her personal accomplishments (no matter how the team performs), as well as the team performance.

Accordingly, in an academic setting, if the course is accompanied by a software development project, we propose to make the project evaluation of each student be based on a personal evaluation (independently of how the team performs), and on an evaluation of the team performance. In order to promote teamwork as well as personal contributions to the project, the two components should be balanced in some way. For this purpose, the students' evaluation is composed of two parts: one the personal component and one the team component. The evaluation of the individual role performance is used for the personal evaluation.

The proposed evaluation scheme, presented in Table 2.2, applies these conclusions. It is composed of an individual component (35%) and a team component (65%); the team component is identical for all members of the team. The main criterion of the individual component of the grading scheme is the personal performance of the student (50%) as well as his or her personal role (25%). The main criterion of the group component is the presentation of the customer stories as well as the time estimations given by the students at each of the three development iterations.

Table 2.2 Grading policy

Individual component (35%)	Group component (65%)
50%—	**60%**—
✓ Weekly reflection	Answer the customer stories and meeting the schedule according to the group time estimations:
✓ Pair programming experience	✓ **(10%)** for iteration 1
✓ Test-Driven-Development exercise	✓ **(25%)** for iteration 2
✓ Weekly presence	✓ **(25%)** for iteration 3
25%—	**25%**—
Performance of a personal role:	Project documentation
✓ Actual implementation	**15%**—
✓ Further development and enhancement	Group evaluation by the academic coach
25%—	
Personal evaluation of the academic coach	

This student evaluation scheme shows that both teamwork and individual contribution count. Accordingly, it is assumed that on the one hand, students will be encouraged to contribute to their team's work, and that on the other hand, those wishing to excel will have the opportunity to improve their grade through the personal component. Consequently, the personal responsibility of each student is increased.

The grading policy presented in Table 2.2 also fosters gradual improvement. This is accomplished by the main parts of each grade component. In the group component, 60% of the grade is achieved by meeting the customer requirements according to the students' own estimations. The first iteration (out of the three iterations) receives the lowest portion (even though it lasts half a semester), thus demonstrating that this iteration is dedicated to learning about the environment, the teammates, the method, and the project. In the individual component, about 75% of the grade is achieved by presence and reflection, for specific exercises, and for performing the personal role. These parts are also characterized by gradual learning. Students are more open to give feedback as the semester proceeds, and role performance is improved after the learning iteration, which is the first iteration of the semester.

Tasks

1. In your opinion, should different kinds of teams be evaluated in different ways?
2. Should teams be formed of students with similar preferences, with different preferences, or according to a specific combination of preferences, with respect to bonus allocation? What considerations should guide each of these options?
3. In what ways is bonus allocation similar to course grading? In what ways is it different?

2.7 Concluding Reflective Questions

1. What were your considerations when choosing a personal role? Why?
2. What were your teammates' considerations when choosing a personal role? Why, in your opinion, were these their considerations?
3. How will you accomplish your role successfully?
4. What are the three main goals of agile software teams?
5. In your opinion, what are the three most important characteristics of agile software teams? How do these characteristics enable them to achieve their goals successfully (Question 4)?
6. In Chapter 9, Trust, we will meet two ideas related to agile teamwork: diversity and ethics. If you are familiar with these concepts, suggest connections between them and the way agile teams are built and function.

2.8 Summary

This chapter introduces the first steps of agile teams by establishing their structure and some of their development habits and processes. This is done by assigning personal roles to team members, which on a personal level improves their understanding of the developed product, and on the team level improves the process and product quality. The role scheme that guides the role assignment achieves these goals by defining personal roles and cross-project roles; that is, each role holder is responsible for the management of a specific aspect throughout the *entire* project.

This chapter also introduces dilemmas in agile teamwork and how they can be addressed. In the spirit of the Agile Manifesto, dilemmas and conflicts associated with teamwork should be discussed openly by all team members, and a solution that meets the needs of the individuals as well as the entire team should be established. This idea has been illustrated by an evaluation scheme for student projects.

References

Dubinsky Y, Hazzan O (2004) Roles in agile software development teams. 5th international conference on extreme programming and agile processes in software engineering. Garmisch-Partenkirchen, Germany, pp 157–165

Dubinsky Y, Hazzan O (2006) Using a role scheme to derive software project quality. J Syst Architect 52 (11): 693–699

Hazzan O (2003) Computer science students' conception of the relationship between reward (grade) and cooperation. 8th annual conference on innovation and technology in computer science education (ITiCSE 2003), Thessaloniki, Greece, pp 178–182

Humphrey W (2000) Introduction to the team software process. Addison-Wesley, Reading, MA

http://www.springer.com/978-1-84800-198-5

Agile Software Engineering

Hazzan, O.; Dubinsky, Y.

2008, VIII, 296 p., Softcover
ISBN: 978-1-84800-198-5