

# Roles in Agile Software Development Teams

Yael Dubinsky<sup>1</sup> and Orit Hazzan<sup>2</sup>

<sup>1</sup> Department of Computer Science, Technion, Israel

<sup>2</sup> Department of Education in Technology & Science, Technion, Israel

**Abstract.** One of the key elements in describing a software development method is the roles that are assigned to the members of the software team. This article describes our experience in assigning roles to students who are involved in the development of software projects, working in Extreme Programming teams. This experience, which is based on 25 such projects, teaches us that a personal role for each teammate increases personal responsibility while maintaining the essence of the software development method. In this paper we discuss ways in which different software development methods address the place of roles in a software development team. We also share our experience in refining role specifications and suggest a way to achieve and measure progress by using the perspective of the different roles.

## 1 Introduction

Agile software development methods (SDMs) are composed of several elements, such as practices, values, roles, techniques, and tools. Different agile SDMs differ in their role specifications. In fact, one way by which an SDM may emphasize its main principles is through the roles that it specifies.

In order to achieve personal responsibility of all teammates when guiding Extreme Programming (XP) projects in the academia, we add personal roles to the original XP roles. By having a personal role, developers are expected to perform their development tasks as well as the tasks related to their personal role. Thus, no teammates are merely developers. As it turns out, the two activities have a mutual positive influence, and consequently, the collaboration between the team members is enhanced. For example, let us assume that one of the teammates is a developer who also has the role of the tester (and as such is in charge of testing activities, such as writing unit tests and guiding other teammates in the writing of tests). This responsibility leads the teammate to write more tests for his or her own development tasks. These tests can, in turn, serve as examples that illustrate to other teammates how unit tests should be written. Another example is when a teammate, who is a developer, also has the role of the customer. On the one hand, telling customer stories leads to an awareness of these stories when developing ones own tasks; on the other hand, the development work may inspire the definition of acceptance tests that are to be defined by the customer. This "changing of hats" is possible as long as everyone is aware of which hat is appropriate for each situation. In other words, each team member plays two

roles and switches between them according to the situation; other teammates comprehend these switches and refer to the appropriate hat depending on the relevant context.

In this paper we elaborate on the roles in a software development team, share our experience in adding roles and refining role specifications, and suggest a way to achieve and measure progress by using the perspectives of these different roles.

## 2 Role Experience

This section describes the evolution of possible XP roles. The description is based on the experience of guiding the development of XP projects in five different one-semester courses, in which 25 projects were developed by about 325 students. Main lessons are highlighted.

**Summer 2002 Semester.** Our first experience with XP projects was in the "Projects in Operating Systems" course given by the Department of Computer Science at the Technion, Israel. The course is a project-based capstone course. Since the Summer 2002 semester, XP has been implemented in the course on a regular basis. The students work in groups of twelve, and each group is guided by an academic coach. Each group has a dedicated equipped Studio (see [4]) for the project purposes.

In the said semester, four XP projects were developed. Every student of every team was required to select one special role out of six possible roles - *assistant coach*, *tracker*, *tester*, *on-site customer*, *release presenter* and *iteration presenter* - and to fill that role for a period of half a semester. The grading policy took the personal XP role into consideration; therefore, each student was required to have a role. In this first semester we decided that the academic coach would play the role of XP coach and would act as the main on-site customer. The role of assistant coach was defined as that of the XP coach but was supervised over by the academic coach. We identified the continuous integration practice as a technical obstacle, especially in an academic environment in which students meet only once a week. Accordingly, during the semester, the *responsibility of continuous integration* was added to that of the release and iteration presenters.

**Lesson 1.** A personal role helps to increase teammates' involvement in and commitment to the software development process.

**Lesson 2.** Role performance improves during the second half of the semester due to the learning that takes place during the first half of the semester.

**Winter 2003 Semester.** In this semester, the second in which the "Projects in Operating Systems" course was offered, we continued with 2 projects, using the same 6 roles used in the Summer 2002 semester. In addition, XP was also introduced, this semester, into a course dealing with operating systems concepts and the teaching of such, which was attended by 30 prospective computer science teachers. The class, working as a single team, developed a single XP project. Roles were assigned in this case as follows: two students were *trackers*; two students were *responsible* for the different stages of the *continuous integration*, and the others were developers.

**Lesson 3.** The academic coach does not have to be the XP coach in order to evaluate the team's work. Therefore, the role of XP coach should be given to a student.

**Lesson 4.** An XP project can be developed by 30 students, but they will not all be involved in the actual development process. In addition, those students who *do* have specific roles tend to feel that they deserve bonus points for their extra work.

**Spring 2003 Semester.** Some changes were made in the roles assigned in the four projects that were developed in the "Projects in Operating Systems" course. The number of roles was increased to seven, since there were some groups with 13 students. The roles were: *coach, assistant coach, tracker, person in charge of continuous integration, tester, person in charge of presentations, and person in charge of documentation*. In this semester, the academic coach was no longer the XP coach. We cancelled the on-site customer student role, assuming that this role would be the focus of the academic coach. We added a documentation role that handles the documentation of the development process, as well as the user's guide and installation manual. We also separated the topic of continuous integration from the presentations.

**Lesson 5.** We realized that we could not do without the on-site customer student role, but that we could give up the assistant coach role since the role of XP coach was now played by one of the students. The appropriate steps were taken at the second release developed later in the semester.

XP was also introduced into two other courses that were held this semester. The first course was on object-oriented concepts and on the teaching of this topic, and was attended by 30 prospective computer science teachers. Special roles were not assigned to the students during this first semester, and the students developed a single project, working as a single team. Similar to our experience the semester before, we found that in this way, too, an XP project can be completed, but again many students were not involved in the actual development process. The second course into which XP was introduced was a course on software engineering methods attended by 22 mathematics major students. In this course, the seven aforementioned roles were assigned, but several roles were performed by more than one student.

**Lesson 6.** The upper limit for the group's size should be about 12 students. The assignment of personal roles solves the problem of lack of involvement in the actual project work.

**Summer 2003 Semester.** During the fourth semester, we had 4 project groups in the "Projects in Operating Systems" course with no more than 12 students in each group. It was in this semester that the list of six roles that we then considered to be an optimal list was reached: *coach, tracker, tester, person in charge of continuous integration, on-site customer, and person in charge of presentations*. The documentation task was added to the role of the team member who was *in charge of presentations*.

**Winter 2004 Semester.** During the fifth semester of the "Projects in Operating Systems" course we again had four project groups and the same list of

six roles was used as in the previous semester. In addition, XP was used in two other courses. The first of the two dealt with operating systems concepts and the teaching of such, and was attended by 18 prospective computer science teachers. The second course was on object-oriented concepts and was attended by 25 mathematics major students. In both courses, the class was divided into two project groups of 9 to 13 students each.

Students were asked to offer topics for projects that were related to the course topics, and then voted on the different subjects until only two subjects remained. From previous experience we had learned that projects that are developed in the framework of courses that are not project-based courses should be based on a single release that is composed of two iterations. Thus, we assigned 13 roles in the largest group; one role per student for the entire duration of the semester. The roles were assigned after several meetings, when the students had become acquainted with each other. Each group was asked to decide on the best way to assign roles to students. This way, each student had a single role to learn, to guide the other teammates accordingly, and to support on-going related activities during the semester.

The roles, on which we will elaborate in the sequel, were *coach*, *tracker*, *person in charge of unit testing*, *person in charge of functional testing*, *person in charge of continuous integration*, *on-site customer*, *person in charge of presentations*, *person in charge of documentation*, *person in charge of design*, *person in charge of code standards and tools*, *end user*, *person in charge of installation shield*, and *person in charge of code correctness and efficiency*.

**Lesson 7.** In the coming semester (Spring 2004), which will be the sixth semester in which we will implement XP in the "Projects in Operating Systems" course, one XP role will be assigned to each student for the entire duration of the semester. This lesson is observed clearly if we examine the learning curve of these roles, and is based on the positive experience expressed in the other courses (see Winter 2004 Semester).

**Note:** In parallel to the above gradual clarification and refinement of the student's roles, the academic coach role was continuously refined as well during the last five semesters. We began by assuming the roles of the team coach and the customer to the academic coach, and underwent several phases through which the responsibility of this role was transferred to the students. A framework for coaching XP projects in the university is presented in [3].

### 3 Roles In XP Teams

In the Appendix<sup>3</sup>, we describe the roles defined by the different agile software development methods [5, 1, 2]. Clearly all agile SDMs have roles that aim to enhance communication and produce a better product. Differences among the methods result mainly from the different emphasis of the SDM itself.

When guiding a software project in the academia, an equal academic load should be assigned to all students. Therefore, according to the number of stu-

---

<sup>3</sup> You may contact [yael@cs.technion.ac.il](mailto:yael@cs.technion.ac.il) for the full version including the appendix.

dents in the project team, some roles are split or, alternatively, several roles are combined into a single role. Indeed, a relevant question that should be asked now is how different roles are split or combined. We have found that *all of the roles together should cover as many as possible of those practices that we wish our students to implement throughout the project development*. The importance of this principle is illustrated by the following example. Teammates may be aware of the importance of continuous integration and may appreciate working at a sustainable pace. These practices may, however, be applied properly (in most of the cases) only if one of the team members actively pushes the team in these directions. Accordingly, we refer to *roles* as *practice representatives*.

In Section 2, we explained the process that led to the formulation of the different roles. In total, we identified 13 roles, which are described and grouped into four major groups in Table 1. The first is the *leading group*, which consists of the coach and tracker. The second is the *customer group*, which consists of three roles. This group of roles focuses on providing the customer with the required product. The third group of roles is the code group, which is composed of five roles and focuses on those aspects of software development that are directly related to the design and to the code. The fourth group is the *maintenance group*, which comprises three roles and focuses mainly on the external presentation of the product. In addition to this grouping, some of the roles support the communications between the four groups. For example, the team member who is in charge of continuous integration is also in charge of communications with the customer group.

## 4 Using Roles to Achieve and Measure Progress

This section presents an analysis of data that was gathered in a qualitative research during the five aforementioned semesters. The data were gathered from videotapes of the meetings of one team in each semester, interviews with students and academic coaches, students' electronic forums and reflections, project presentations, and the impressions and periodical summaries of the various role holders. This data helps us illustrate how roles can be used to achieve and measure the progress of the software project.

The progress is examined from the following three perspectives: endowing XP values, learning XP practices, and increasing awareness to the human aspects of software development. Measurement of progress using roles is executed by examining the adherence to the time schedule and to the customer stories.

We found that the XP values establish a valuable framework for teamwork. Having a role causes each teammate to become more involved and much more communicative with other team members. For example, it is not possible to motivate one's teammates to write unit tests or to write according to specific coding standards without extensively communicating with them. Courage is required in order to take on additional responsibility besides being a developer, to accomplish the required work and to urge the other teammates to follow one's instructions within a specific area of responsibility. Feedback is provided

**Table 1.** Roles in an academic XP team

Role	Description
<b>Leading Group</b>	
Coach	Coordinates and solves group problems, checks the web forum and responds on a daily basis, leads some development sessions.
Tracker	Manages the group diary, measures the group progress with respect to the estimations and tests score, manages and updates the boards.
<b>Customer Group</b>	
End user	Performs on-going testing of the software as an end user, contacts real end users to test the software, collects and processes the feedback received.
On site customer	Tells customer stories, makes decisions pertaining to each release and iteration, provides feedback, defines and develops acceptance tests.
In charge of acceptance testing	Works with the customer to define and develop acceptance tests, learns the topic of first-test development and instructs the others on the subject.
<b>Code Group</b>	
In charge of unit testing	Learns about unit testing, establishes an automated test suite, guides and supports others in developing unit tests.
In charge of design	Maintains current design, works to simplify design, searches for locations in the software that need refactoring and ensures proper execution of such.
In charge of code standards and tools	Establishes and refines group code standards, searches for development tools that can help the team, guides and supports in the maintaining of standards and use of tools.
In charge of code effectiveness and correctness	Guides other teammates in the benefits of pair programming, enforces code inspection in pairs, searches for places in the code whose effectiveness requires improvement.
In charge of continuous integration	Establishes an integration environment including source control mechanism, publishes rules pertaining to the addition of new code using the test suite, guides and supports other teammates in the integration task.
<b>Maintenance Group</b>	
In charge of presentations	Plans, organizes and presents version presentations, demos, and time schedule allocations.
In charge of documentation	Plans, organizes and presents the project documentation: process documentation, user's guide, and installation instructions.
In charge of installation shield	Plans and develops an automated installation kit, supports and instructs other teammates as to the appropriate way to develop software for easy and correct installation.

to others and received from other's concerning one's role and performance. In turn, this feedback increases communication. When assuming responsibility for a specific topic related to the development of a software project, one wants it to be as simple as possible in order to easily establish and maintain it. Simplicity naturally leads to the assuming of the appropriate scope of one's responsibility. Table 2 presents students' feeling about their roles with respect to XP values.

**Table 2.** Students' feeling about their roles with respect to XP values

Role	XP Values	Students' expressions
Tracker	Simplicity	We do it the simplest way because we have tons of other things to do and we aren't looking for unnecessary complications.
Coach	Courage	First of all, I don't have the characteristics of a manager; I'm quite shy, not charismatic...
In charge of continuous integration	Communication	It's also hard to urge everyone to do their part ... All of this is of course understandable, and I believe I handled it well ... together with the hard work on the project.
In charge of documentation	Communication and feedback	This role is recommended for people who like to interact with other people, whether if it's in the presentation, the making of the presentation, the coding documentation or the project's working process report. If you don't like these things too much - take another role. If you do, I recommend ... Make sure from the start that people ... It is very important that they get used to doing it during the entire process of coding, and not just at the end, because ... Pay attention to the fact that people are used to ... so you have to be tough...
Customer	Feedback	As a customer, I wrote customer stories and decided... and gave feedback.
Tester	Communication	I continuously pushed them and asked them to write testing for their units and publish it on the forum. The main problem was that some of the team members didn't finish ... and in some cases I asked the coach for help in obtaining the test code.
Coach	Courage	I would say that a substantial part of the coach's duty was rendered superfluous due to the effort made by the entire group to work as a team.

The need to learn the XP practices leads to an on-going refinement of role definitions. Students performed their roles while learning the XP practices. Grad-

ually, they became practitioners. Following are students' expressions of their feelings with respect to their perception of the different XP practices.

**Customer:** I had to follow and see that during implementation time people were working according to my stories.

**In charge of unit testing:** I published two documents that explain the testing subject. I published a request that teammates send me their planned tests for each module... I gave a short lecture about software testing...

**Coach:** I provided the team with the applications and operating systems, I tried to coordinate and make people move fast...

**In charge of documentation:** I published a documentation guidelines that also deals with coding techniques, and checked the team's code to see if they played along.

The human aspect of software development is a broad area. In this paper, we focus on students' feelings and awareness with respect to their roles, as expressed by them during the development process. Satisfaction on the part of the students in being role holders was observed, as well as in being able to obtain a global view of the project in addition to the accomplishment of specific development tasks. Most of the students reported that they handled this additional responsibility well and enjoyed it. Following are students' expressions of their feelings about their role handling.

**Customer:** The role gave me a "real life" feeling, not that we have a predefined task and we just perform it. This is very real, a customer with requirements,...

**In charge of continuous integration:** I enjoyed seeing that everything was integrated...

**In charge of unit testing:** I didn't enjoy the role at all ... it caused me a great deal of nervousness in the past two months...

**In charge of documentation:** So *I* wrote the documentation that *he* was supposed to write...it didn't kill me, but I consider it as a personal failure.

Measuring the development progress is usually a complicated task. As it turns out, by using roles we can obtain information on many of the elements of the progress of a software project in the form of narratives expressed by role holders. We used three narrative tools: stand-up meetings, periodical summaries by roles holders, and role holders' web expressions and reflections. An analyzed collection of the narratives information at every stage gives quick glances on the status of the team, and when looking at them over time, the progress in the different aspects of the project is revealed. Following are quotes taken from role holders' summaries of a specific project. These summaries were written at the beginning of the project after one week of development and two weeks before the presentation of the first iteration of the first release.

**End user:** I worked with the customer. We met with the coach in order to discuss the graphical interface. We defined each button...

**Coach:** I met most of the teammates in order to coordinate... I worked with the tracker on the documentation and publishing of the development tasks...



**In charge of installation shield:** I'm going to search for installation software and try to learn it for future use.

**In charge of unit testing:** I learnt about the subject...

**In charge of presentations:** For now, no actions concerning my role were required, but there soon will be.

## 5 Conclusion

It is a well-known fact that software development is a complicated process. In practice, a very unique kind of teamwork is required in order to accomplish its many significant elements. This paper raises the question whether each teammate in a development team should have one major role in addition to his or her personal development tasks. It is suggested that when a teammate has a specific role, his or her personal responsibility and accountability with respect to that aspect of the software development process represented by the said role, increase. The total array of roles enables the accomplishment of all practices we wish to include in the development process and leads to a high involvement of all teammates in the development process. Although this article presents data analysis of XP projects conducted in a university setting, we suggest that the above conclusion need not be limited to the academia, but rather its implementations for the software industry should be considered as well.

## 6 Acknowledgements

This research was supported by Technion V.P.R. Fund - B. and G. Greenberg Research Fund (Ottawa).

## References

1. Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley 2000.
2. Crispin, L. and House, T.: Testing Extreme Programming. Addison-Wesley 2002.
3. Dubinsky, Y. and Hazzan, O.: eXtreme Programming as a Framework for Student-Project Coaching in Computer Science Capstone Courses. Proceedings of the IEEE Int. Conf. on Software - Science, Technology & Engineering, pp. 53-59, 2003.
4. Hazzan, O.: The reflective practitioner perspective in software engineering education. The Journal of Systems and Software 63(3), pp. 161-171, 2002.
5. Highsmith, J.: Agile Software developments Ecosystems. Addison-Wesley 2002.

## APPENDIX: ROLES IN AGILE SOFTWARE DEVELOPMENT METHODS

SDM	SDM Roles
XP (Beck, 2000)	<p><b><u>Seven roles in Extreme Programming (XP):</u></b></p> <p>The <b>programmer</b> analyzes, designs, tests, programs, and integrates. The programmer writes tests and refractors the code; working as part of a pair. The programmer communicates and coordinates closely with other programmers in order to ensure the project's success. The programmer integrates the code and shares it with the others.</p> <p>The <b>customer</b> tells and writes stories to be implemented and decides when they will be implemented. The customer defines tests to verify the correct functionality of the stories. The customer receives feedback from the team, and makes decisions to help the team best benefit the project.</p> <p>The <b>tester</b> uses the customer's viewpoint in order to determine which items most require verification. The tester must consider the system through the eyes of the customer (Crispin and House, 2002).</p> <p>The <b>tracker</b> measures progress quantitatively, by comparing estimations with actual results. He or she is responsible for monitoring the big picture and informing the teammates about their progress. The tracker is the team historian, and keeps log of tests results and reported faults/defects.</p> <p>The <b>coach</b> is responsible for the process as a whole. He or she keeps track of the project's process and helps other teammates in their decision making. The coach pairs off with programmers, identifies/looks for refactoring tasks, and sees to their execution. The coach also explains the process to upper-level managers.</p> <p>The roles of <b>consultant</b> and <b>boss</b> are external and are filled by people from outside the team.</p>
DSDM	<p><b><u>Eleven roles in Dynamic Systems Development Method (DSDM):</u></b></p> <p>The <b>executive sponsor</b> is a high-level executive who is responsible for the system and for its fast development progress.</p> <p>The <b>ambassador user</b> represents the entire user community.</p> <p>The <b>visionary user</b> makes sure that the vision of the product is not lost.</p> <p>The <b>advisor user</b> brings daily business knowledge to the development team.</p> <p>The <b>project manager</b> is responsible for ensuring project delivery, coordinating and reporting to the management.</p> <p>The <b>technical coordinator</b> reports to the project manager and assists all development teams.</p> <p>The <b>team leader</b> ensures that the team functions as a whole, and that the objectives are met.</p> <p>The <b>senior developer</b> interprets user requirements into prototypes and deliverable code.</p> <p>The <b>developer</b> assists with these tasks as part of DSDM skills development.</p> <p>The <b>facilitator</b> is responsible for managing the workshop process, an interactive communication technique for making decisions.</p> <p>The <b>scribe</b> records requirements, agreements and decisions reached.</p>
Scrum	<p><b><u>Four roles in Scrum:</u></b></p> <p>The <b>scrum master</b> reviews the team's progress team and ensures time estimations are</p>

	<p>updated.</p> <p>The <b>product owner</b> writes user stories and defines acceptance tests.</p> <p>The <b>scrum team</b> estimates task durations and develops stories and unit tests.</p> <p>The <b>manager</b> provides directions to keep the work going according to plan and removes obstacles.</p>
Crystal Clear	<p><b><u>Eight roles in Crystal Clear:</u></b></p> <p>Distinct roles:</p> <p>The <b>sponsor</b> provides the mission statement.</p> <p>The <b>senior designer</b> produces the system design.</p> <p>The <b>user</b> helps with use cases and screen drafts.</p> <p>The <b>designer-programmers (designers)</b> design, code and test.</p> <p>Four additional merged roles are identified in Crystal Clear, which means that they can come from the people filling the above-mentioned roles:</p> <p>The <b>business expert</b> can come from the sponsor, user, or senior designer.</p> <p>The <b>coordinator</b> can come from the senior designer and is responsible for the schedule and the release sequence.</p> <p>The <b>tester</b> can come from the designers and is responsible for test results and defect reports.</p> <p>The <b>writer</b> can come from the designers and is responsible for the user manual.</p>
FDD	<p><b><u>Six [core] roles in Feature-Driven Development (FDD):</u></b></p> <p>The <b>project manager</b> leads the team and reports on its progress.</p> <p>The <b>chief architect</b> is responsible for system design.</p> <p>The <b>development manager</b> is responsible for the development activities.</p> <p>The <b>chief programmers</b> provide technical leadership to the smaller teams.</p> <p>The <b>class owners</b> are developers who each own one class and are responsible for making all changes in it.</p> <p>The <b>domain experts</b> are the users.</p>
Lean Development	<p><b><u>Six roles in Lean Development:</u></b></p> <p>The <b>customer</b> provides the requirements.</p> <p>The <b>master developer</b> is responsible for system design.</p> <p>The <b>expertise leader</b> is responsible for specific technical areas such as GUI design, database development, and security.</p> <p>The <b>project leader</b> is responsible for time estimations and the team's progress.</p> <p>The <b>observer</b> takes notes on the team's process.</p> <p>The other team members are the <b>programmers</b>.</p>
ASD	<p>Adaptive Software Development (ASD) promotes the leadership-collaboration model, which focuses on work states rather than on processes, on creating a collaborative environment, and on creating accountability for results (Highsmith, 2002).</p> <p><b><u>Six roles are mentioned in ASD:</u></b></p>

	<p>The <b>executive sponsor</b> is responsible for the product being developed.</p> <p>The <b>developer</b> and <b>customer</b> representatives.</p> <p>The <b>facilitator</b> plans and leads the development sessions.</p> <p>The <b>project manager is</b> responsible for product delivery.</p> <p>The <b>scribe</b> records requirements, agreements and decisions reached.</p>
--	--